VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Vo Thanh Vinh

AUTONOMOUS PACKAGING ROBOT

Technology and Communication

2010

PREFACE

The application presented in this paper has been done at the Telecommunication and Information Technology Department in the Vaasa University of Applied Sciences from May 2010 to September 2010.

First, I would like to thank my supervisor, lecturer Yang Liu, for giving me whole-hearted assistance and guidance to complete successfully my thesis. He has been giving me advices that can be considered as vital factors for the success of the application.

I would like to express my gratitude to Prof. Petri Helo for bringing this idea to light. Prof. Petri Helo is the first one to understand the need for this application in the real life. The idea of combining robot, vision and computer together is first started by him. He also provided me with all hardware components needed to complete this application. Therefore, to him I am grateful.

Many thanks go to Dr. Smail Menani and the lecturer Mika Billing for their continuous help and advice. They have created a flexible and dedicated working environment inside Vaasa University of Applied Sciences and Technobothnia.

I would like to credit all members of the Telecommunication and Information Technology Department in the Vaasa University of Applied Sciences for maintaining the high quality and comfortable, learning-oriented environment during my period of education.

Vaasa, 4 October 2010

Vo Thanh Vinh

VAASAN AMMATTIKORKEAKOULU

UNIVERSITY OF APPLIED SCIENCES

Degree Programme of Software Engineering.

ABSTRACT

Author Vo Thanh Vinh

Title Autonomous Packaging Robot

Year 2010

Language English

Pages 110

Name of Supervisor Yang Liu

The objective of the autonomous packaging robot application is to replace manual product packaging in food industry with a fully automatic robot. The objective is achieved by using the combination of machine vision, central computer, sensors, microcontroller and a typical ABB robot.

The method is to equip the robot with different sensors: camera as “eyes” of robot, distance sensor and microcontroller as “sense of touch” of the robot, central computer as “brain” of the robot. Because the robot has its own “hand” and “senses”, this implementation will enable robot to work in factories without human interference.

The application implementation is presented in this paper. The proposed method is stable and robust in the testing environment. Practically, the result has been evaluated as a success in both precision and timing.

Keywords Robot, Packaging, Vision, Range Finder, ABB SC4

CONTENTS

1 INTRODUCTION

1.1 Background

The initial aim of this application is to raise the efficiency in food industry. However, it can be expanded to be used in other fields as well.

An essential task in a food company is to pack products in a container. The customer places orders with the company and operators will input order information to the system. Those orders are exposed to factories in form of web service. In the factory, each worker has a working area equipped with a computer which has a dedicated client program connected to the web service. There are two types of box: input box and output box. The worker will receive the information from the computer's screen and pick products from the input box to the output box according to the information on the computer screen.

Figure 1. Manual workers.

Figure 2. Manual workers.

Figure 3. Typical container box: 537 mm x 337 mm x 234 mm.

1.2 Objective of the application

This project apart from promoting academic intellectual accomplishment, it has real impact in production process nowadays, if it is applied successfully. Manual labour can be replaced by precise, long-lasting and cheap robotic technology. The initial application's target is for food packaging industry where workers have to work 2-3 shifts per day just to pick and place foodstuff from one container box to another according to customer's orders.

Figure 4. New packing solution with robot, vision and computer.

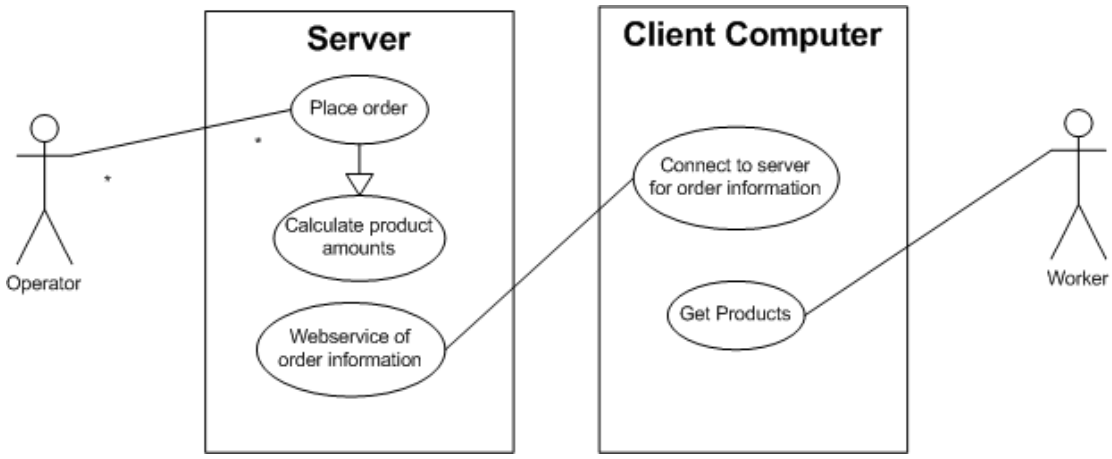


Figure 5. Current Use Case Diagram in the factory.

Figure 5 represents the user diagram of current activity in the factory with manual workers working with a computer screen in front of them.

The perspective of replacing this whole manual process is realistic in current technological advancement. Development plan is described in following user diagram

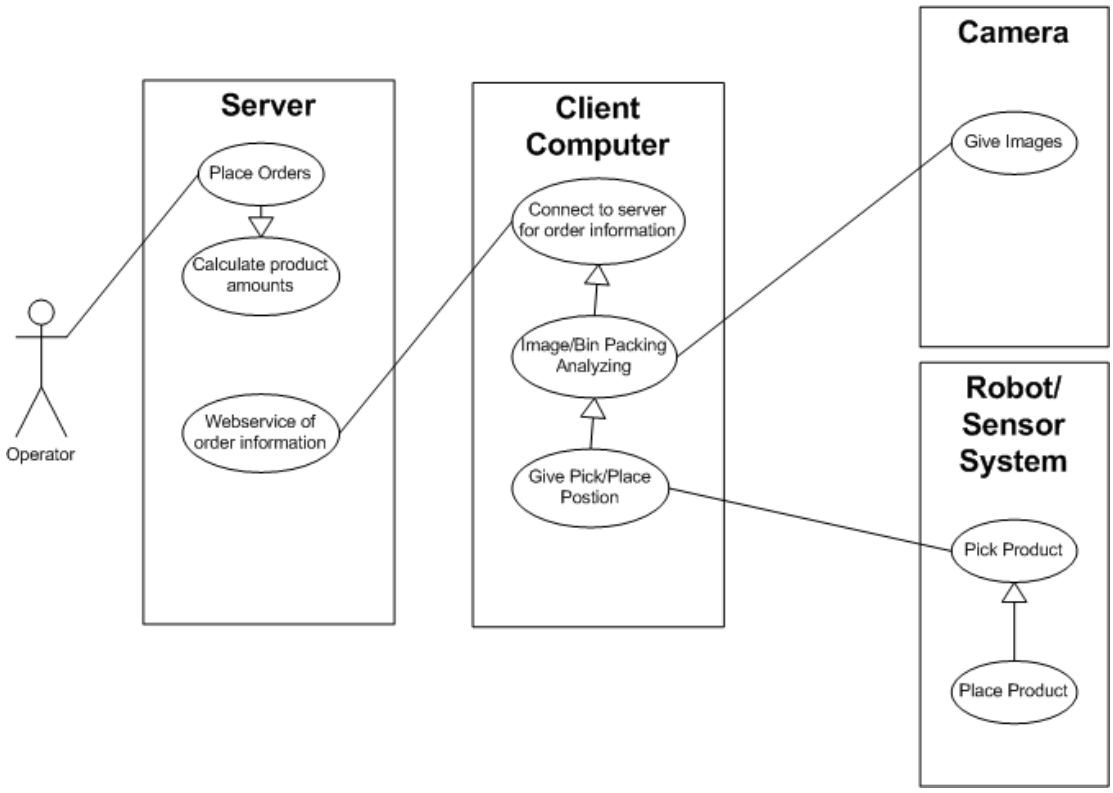


Figure 6. Use Case Diagram of the application.

From above user diagram, the whole factory process is automated with the help of a human-being operator to input customer's orders.

1.3 Summary

In this chapter, overall objective and implementation plan has been described. In the chapters to follow, different techniques and programming languages used to achieve the objectives of the Autonomous Robot Packaging Robot application will be described in details.

2 SYSTEM OVERVIEW

2.1 Introduction

In this chapter, the overview of application’s design is presented. Before starting to perform actual implementation, current technology potential has been perused: what can and can’t be achieved by current technology in order to come up with a realistic design.

2.2 System Design

Below is the block diagram of the application which represents all separate hardware parts of the system.

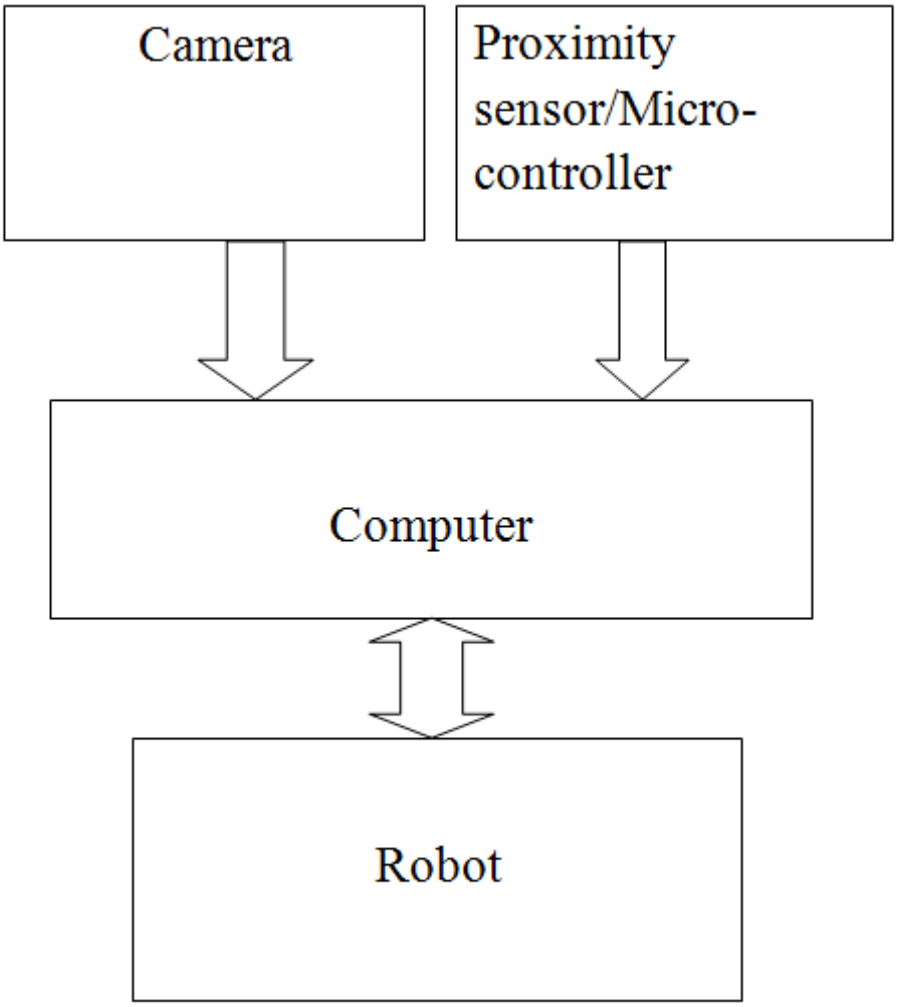


Figure 7. Block Diagram.

The system consists of 4 main parts: camera, proximity sensor system, central computer and robot. The above design will be used to enable the following sequence diagram.

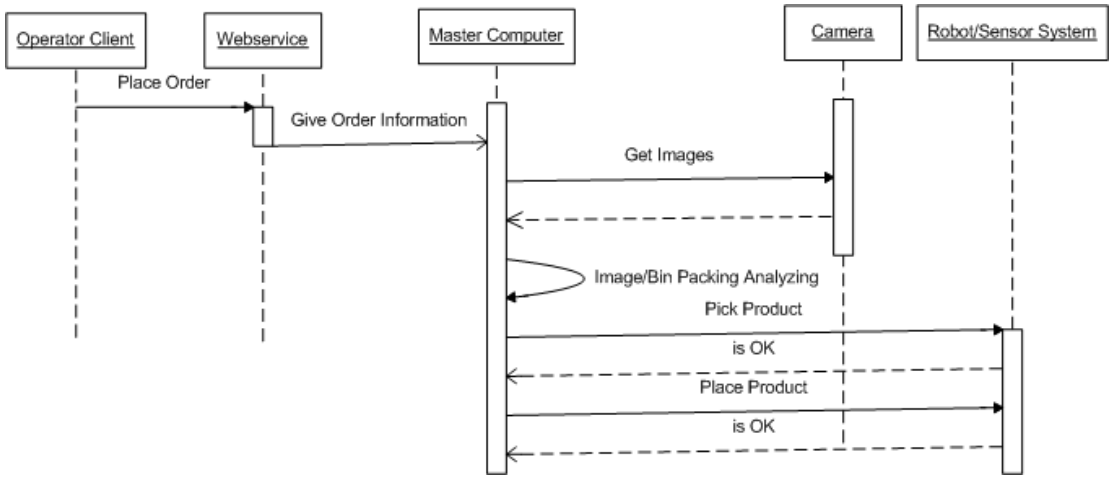


Figure 8. Sequence Diagram.

According to the sequence diagram the whole packaging process is automated completely without the need of human interference.

The activity diagram will give a broad view of the application's flow.

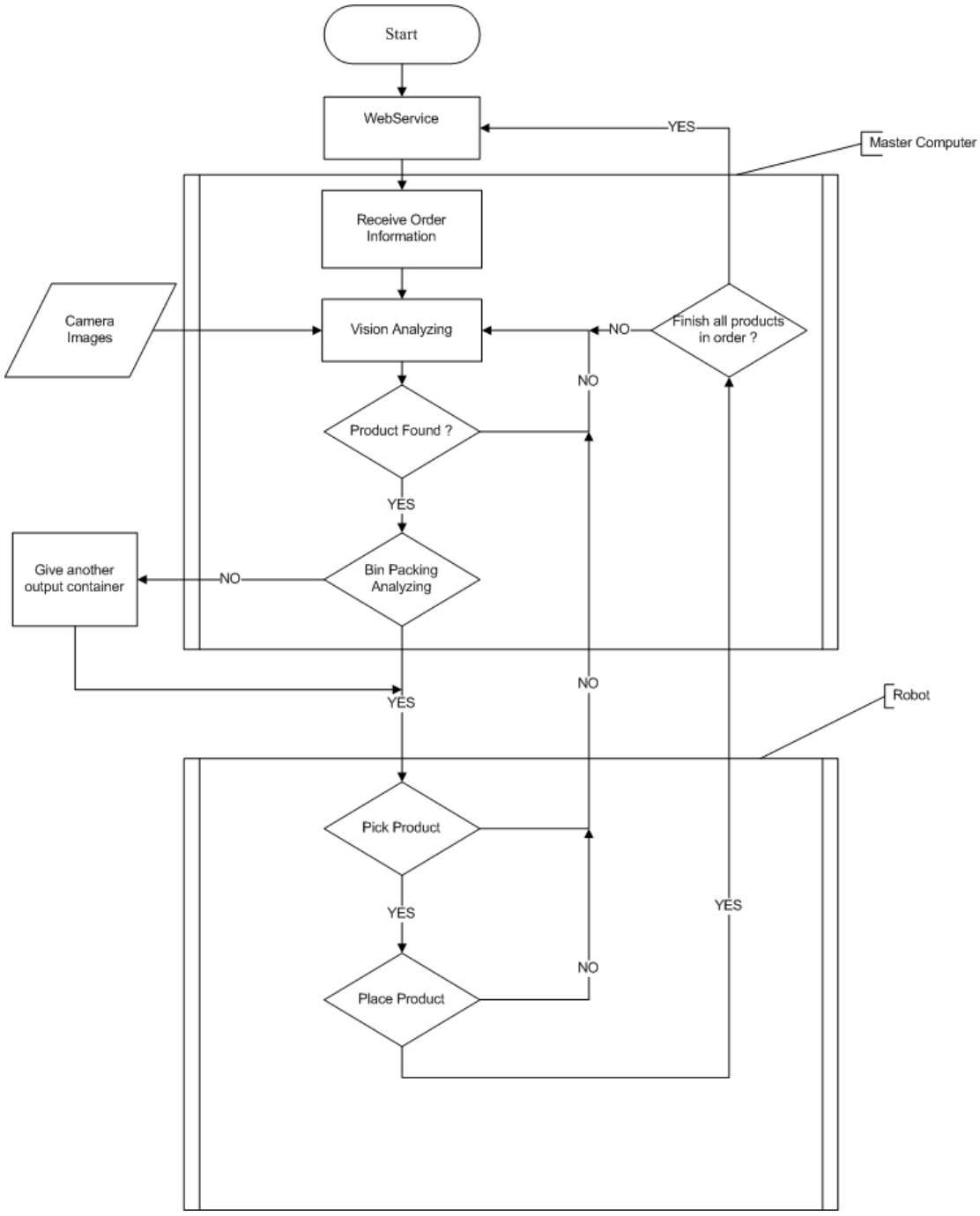


Figure 9. Activity Diagram.

2.3 Summary

All diagrams in this chapter have represented the application's target in a more engineering-friendly manner. Moreover, web service part will not be implemented because it is considered as minor and non-technologically challenging problem. Therefore, the web service part will be replaced by a simple part that generates bogus information of customer's orders.

3 ABB ROBOT

3.1 Introduction

The ABB Group is a leading supplier of robot in manufacturing systems and services: automotive, cement, mineral & mining, marine solution … ABB robot is equipped with advanced mechanical configuration as well as robot controller software. .Therefore, performance including speed control, accuracy position, programmability and communication with external devices is ensured.

Some important properties of SC4 ABB robot

- S4C controller: This relatively new controller software released by ABB (latest version is IRC5) is a compact controller that deliver high-performance.S4C comes with QuickMove&TrueMove functions. QuickMove ensures the highest acceleration of robot's axes. TrueMove takes care of the accuracy position of robot's axes.

Figure 10. S4C Controller [32].

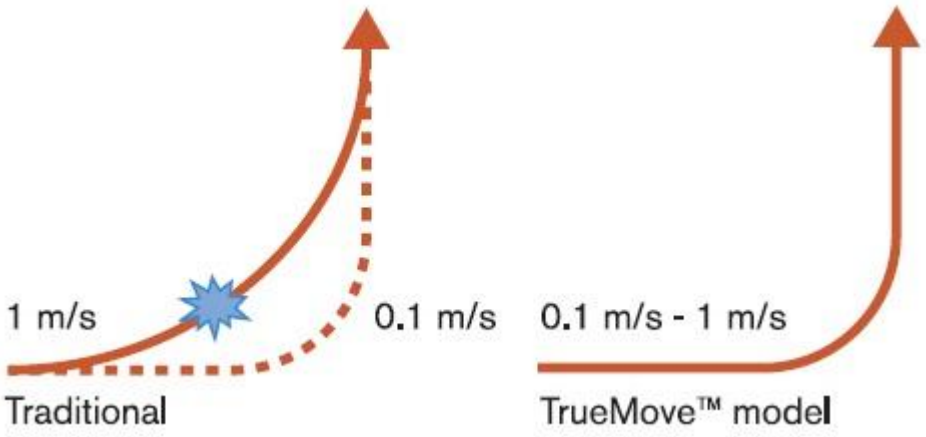


Figure 11. Comparison between Traditional and TrueMove model [33].

- ABB Rapid programming: The 4.0 ABB Rapid which is a C-based programming language comes along with the controller.
- I/O-System: The robot is equipped with some of the standard communication protocols: multiple discrete I/O channels, fieldbus channel, RS232. Those will allow robot to communicate effectively with peripheral devices.

Figure 12. SC4 robot in Laboratory.

3.2 Implementation

In the design of the system, a computer plays central role- the master. Whereas, robot will be considered as a slave which listens to the commands from the computer. Robot and computer are linked to each other by using a RS232 link.

The robot will typically receive following information from the master computer

- X, Y, Z coordinates of the product.
- Rotation of the product.
- X, Y, Z destination coordinates of the product.

The robot will use received information to navigate precisely to pick and place product in right places. Moreover, the robot also plays very active role by asking for new information if it has finished previous job.

Below is the main function in the robot controller

```
 PROC main()
myspeed.v_ori:=2000;!set custom rotation speed
Release; !reset the vacuum grasper
SingArea\Wrist; !mornitor axis configuration at the stop point.
MoveJ home_1,myspeed,fine,Imukalu\WObj:=wobj_use; !move to starting
position
WaitTime\InPos,0; ! wait until robot is in place.
originalOrient:=home_1.rot;
Open comPortName,comport\Bin;! Open COM port
AskMore; !Ask computer for more product to pick / place
    WHILE TRUE DO
        info:=ReadStrBin (comport,MESS_LEN); ! read a line from comport ,
this will wait until !it gets a '\n',30 min time out
         GetTarget; !parse message from computer to usable variables
         IF isOKTHEN !check if data is recievedsucessfully
                  MovePickUp; !pick up product
                  WriteStrBincomport,isReady; ! Ask the computer if the
pickup process has been doing successfully(the computer will read value
from the proximity sensor and decide)
                  isReadySignal:=ReadStrBin (comport,VALUE_LEN); ! read a
line from !comport , this will wait until it gets a '\n',30 min time out
                  TPWrite(isReadySignal);
                  IF StrMatch(isReadySignal,1,isGrasped)
<StrLen(isReadySignal) THEN !if !the pickup has been executed
successfully, procede to place product.
                        MoveToTarget;!place product
                        WriteStrBincomport,leaveCommand;!confirm left
                        MoveBack;!move back to home
                        ClearIOBuff comport; !clear com port data
                        AskMore; !when finish the job, ask more more job
.....
                  ELSE !if the vacuum grasper has failed to grasp product
                     Release; !reset the vacuum grasper
                  ClearIOBuff comport; !clear com port data
                  AskAgain; !tell the computer that : it failed , give me
again
                  ENDIF

            ENDIF
             ENDIF
    ENDWHILE
  ERROR
   ! time out handler
    IF ERRNO=ERR_DEV_MAXTIME THEN
TPWrite "Time out";
    ENDIF
```

The main function described the routine of the robot as follow

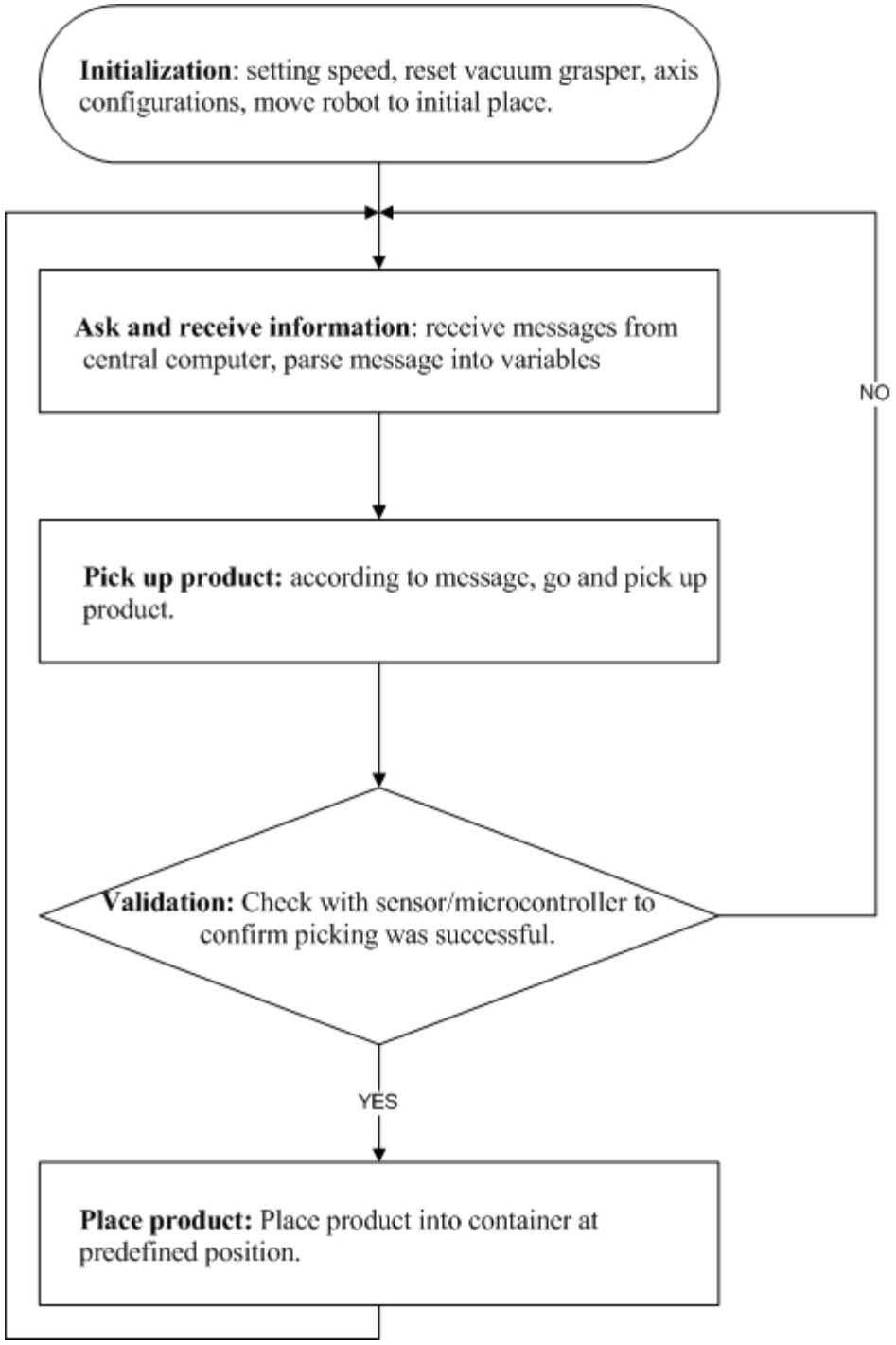


Figure 13. Robot's routine.

3.3 Summary

In this chapter, an overview about ABB robot is presented both in mechanic and programmability. Moreover, details in implementation are described in RAPID code and diagram as well. The robot's implementation has a strong bond with the communication part (sending and receiving messages) which will be presented in chapter 7 "Communication System".

4 VISION SYSTEM

4.1 Introduction

Vision system is a system that actually can see for specific purpose. In particular, in this topic, it is machine that is able to see and perceive position, pose and type of products in container box. In this chapter, all necessary steps of calibration and image processing will be presented.

EmguCV has been brought in use to in order to utilize image processing functions from OpenCV in .Net platform.

4.2 Explanation and Implementation

4.2.1 Camera Model

Pin-hole model is a simple and useful way used to represent a camera. In this model, light is visualized as starting from an object in the scene. Any particular point in surface of object corresponds to exactly one light ray entering the pin-hole. The light ray, then, ends up intersecting with a plane (image plane or projective plane). The intersecting point in image plane is the image representation of a particular point on the surface of that object.

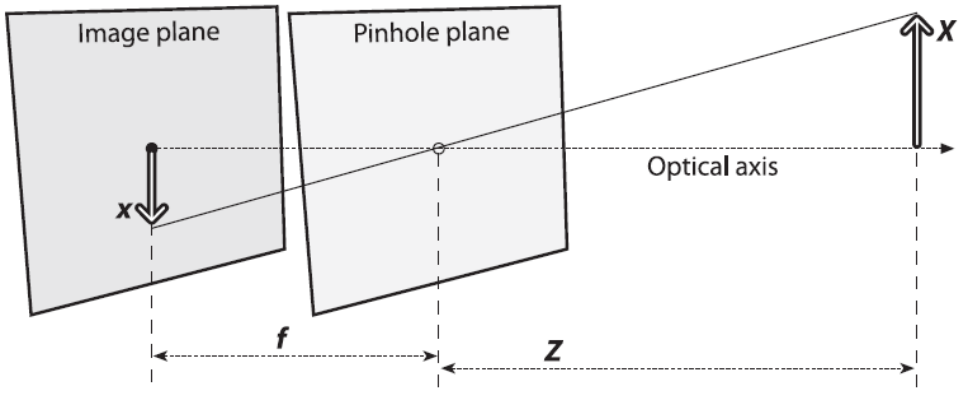


Figure 14. Pinhole camera model [2].

The pin-hole model is mathematically represented by formula

$$-x = f * \frac{X}{Z} \quad \textbf{(1)}$$

where

f is the focal length of the camera.

Z is the distance from camera to the object's point.

X is the height of the object's point from optical axis.

x is the is the image point of the object's point.

This representation is very straightforward for intuition. However, one more step can be used to simplify the mathematic representation by swapping the pinhole plane and image plane.

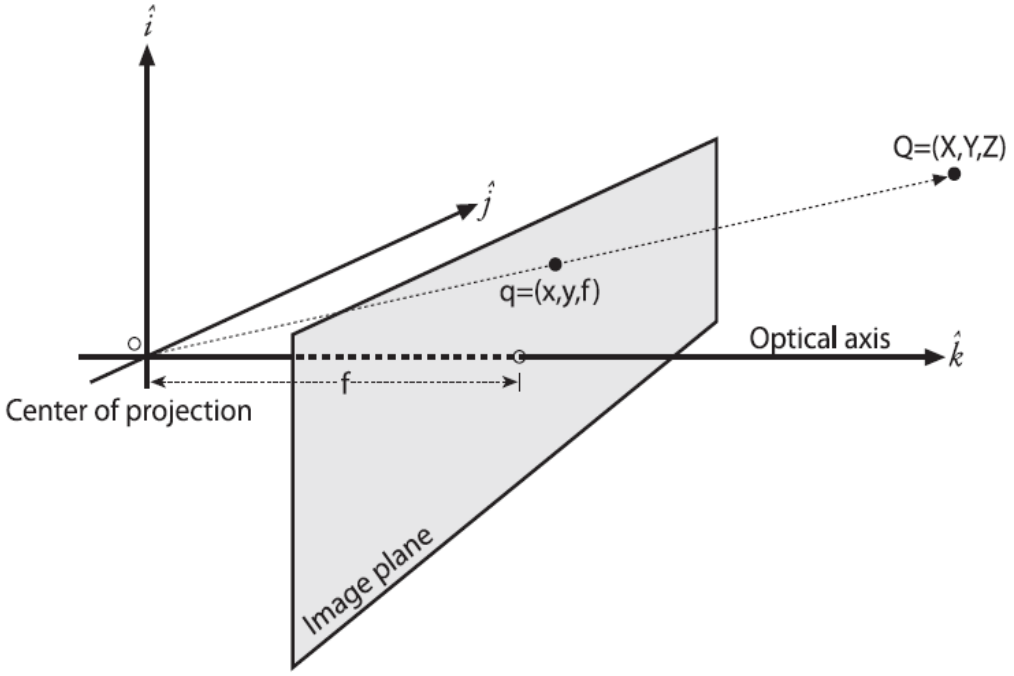


Figure 15. Mathematically simplified pinhole model [2].

In this new arrangement the pinhole point is reinterpreted as centre of projection. Therefore, a simpler triangle relationship is used

$$x = f * \frac{X}{Z} \qquad \textbf{(2)}$$

The negative sign is got rid of because the object image is no longer inverted upside down.

In addition, there are two fundamental problems if simple hole model is applied to the camera

- It is practically impossible to attach the image plane with its centre right on the optical axis to the camera.
- The physical focal length is fixed in the unit of meter. However, the unit of pixel is used in image processing. The focal length is the product of physical focal length (in millimetre) and the size of an individual imager element (in

pixel per millimetre). Again, there is no imager can be produced with a perfect square pixel but rectangle instead. As a consequence, focal lengths in x and y axis are not identical.

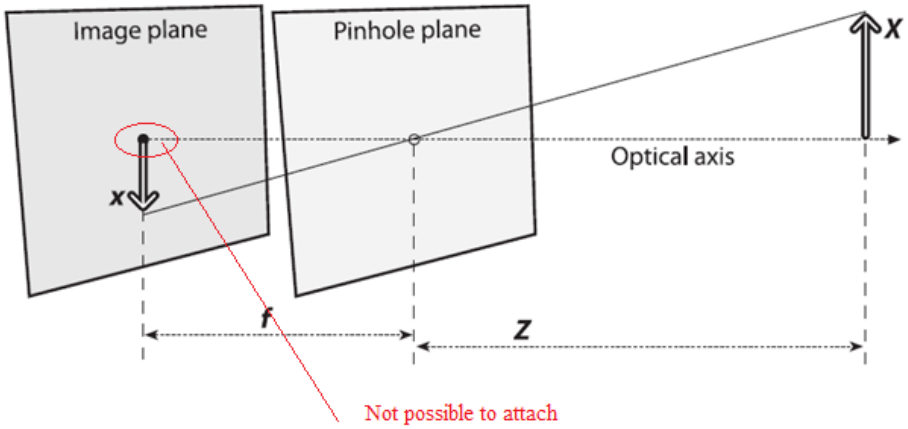


Figure 16. Difficulty in attachment of image plane [2].

In order to cope with mentioned problems, new formulas are introduced

$$x = f_x * \left(\frac{X}{Z}\right) + c_x \quad \textbf{(3)}$$

$$y = f_y * \left(\frac{Y}{Z}\right) + c_y \quad \textbf{(4)}$$

where

f_x *is* the focal length of the camera in x axis.

f_y *is* the focal length of the camera in y axis

Z is the distance from camera to the object's point.

X is the x coordinates of the object's point.

Y is the y coordinates of the object's point.

x is the is the image point's x coordinate of the object's point.

y is the is the image point's x coordinate of the object's point.

c_x is the x coordinate of the centre of image plane.

c_y is the y coordinate of the centre of image plane.

Two focal lengths (f_x and f_y) are used to overcome the problem of not-square pixel. In addition, two new parameters (c_x and c_y) are brought in use to get over the problem of imperfect attachment of image plane.

In summary, two above formulas which represent mathematically the projection of the points in the physical world into the camera can be put in a simple matrix formula

$$s*m'=A*M' \quad \textbf{(5)}$$

or

$$\begin{bmatrix} x \\ y \\ Z \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \textbf{(6)}$$

where A is named camera intrinsic parameter matrix[1].

The drawback of pinhole model is the acquiring image speed. Due to the small size of the pinhole, very little light is collected. As a result, it takes long time to accumulate enough light to construct a complete image of a scene. Therefore, lenses are used to

focus a large amount of light on the pinhole of the camera in order to achieve faster speed.

Unfortunately, no lens is perfect. The main reason is that it is not possible to produce an ideal parabolic lens and align them exactly on camera's focal axis. Therefore, lens always has radial distortions for its imperfect shape and tangential distortions for its imperfect instalment.

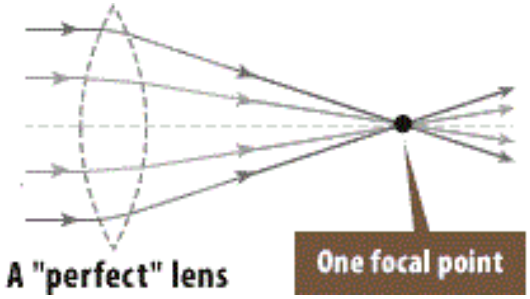


Figure 17. Perfect Lens [2].

Radial distortion (sometimes called edge or rear distortion) is the phenomenon in which as the rays get closer to the edge of lens, they are bent more (fish-eye effect). Tangential distortion is caused by its unparallel assembly with the image plane.

To compensate those defects, new parameters named distortion parameters are added to simple pinhole model [2][3]. New formula is introduced [4]

$$x' = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 xy + p_2(r^2 + 2x^2) \quad \textbf{(7)}$$

$$y' = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_2 xy + p_1(r^2 + 2y^2) \quad \textbf{(8)}$$

where k_1, k_2, k_3 are radical distortion coefficient; p_1, p_2 are tangential distortion coefficients.

Therefore, the first purpose of camera calibration is to find camera intrinsic matrix and distortion parameters. This step is called intrinsic calibration and only need to be done once because those parameters are fixed per each camera.

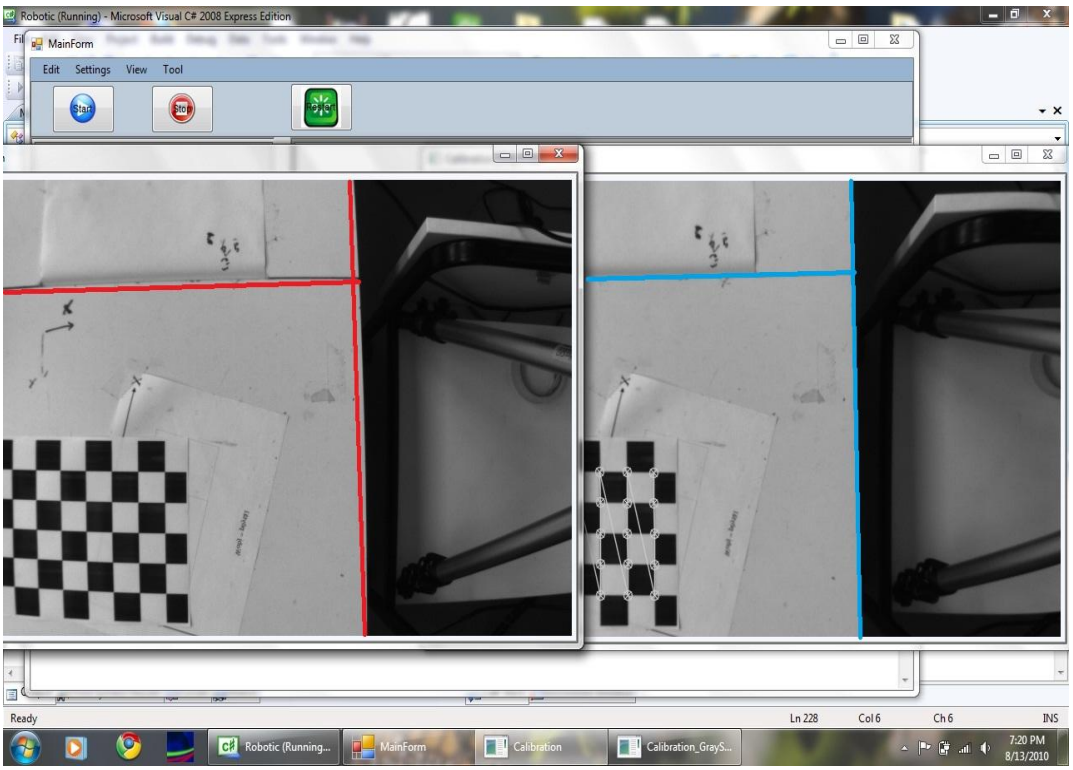


Figure 18. The effect of distortion. Left side and right side are images captured from the same camera at the same time in the laboratory. The left image is suffered from distortion with curved lines. The right image has been calibrated to eliminate unexpected effects.

Usually, the world coordinate system is not overlapped with the camera coordinate system which means that it involves rotations and offsets

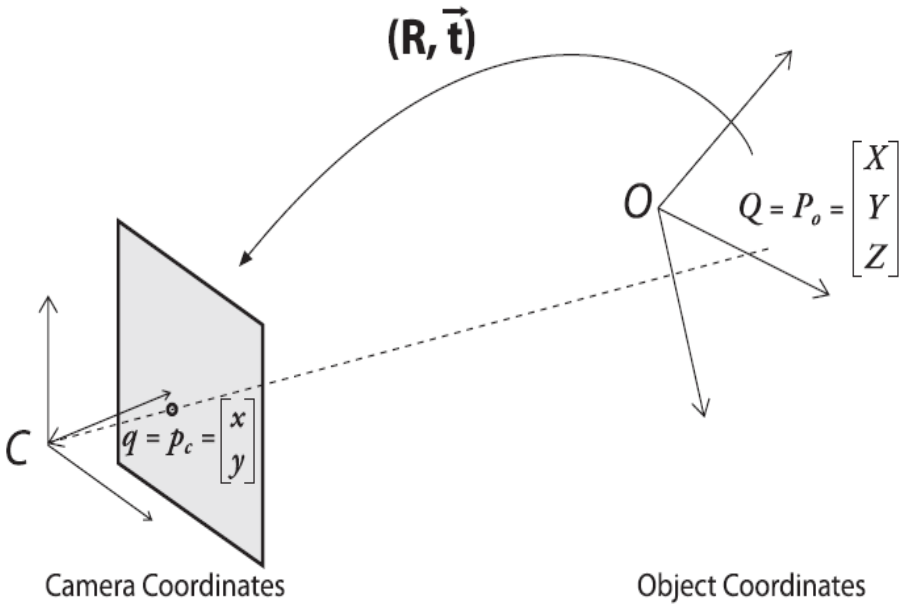


Figure 19. Camera Coordinate and Object Coordinate [2].

Therefore, new matrixes are brought in use to transform from the object coordinate system to camera coordinate system

$$M' = [R|t].M \quad \textbf{(9)}$$

or

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \textbf{(10)}$$

or

$$M' = R*M+t \quad \textbf{(11)}$$

or

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t1 \\ t2 \\ t3 \end{bmatrix} \quad \textbf{(12)}$$

where

R is rotation matrix (to handle rotation between two coordinate systems)

t is translation vector (to handle the offset between two coordinate systems' origin).

As the result camera model can be summarized into one simple mathematic representation

$$sm'=A[R|t]M \quad \textbf{(13)}$$

or

$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad \textbf{(14)}$$

Where

$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$ is the coordinate of a point in the object coordinate system.

$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ is the rotation matrix

$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$ is camera intrinsic matrix.

$\begin{bmatrix} t1 \\ t2 \\ t3 \end{bmatrix}$ is the translation vector.

$\begin{bmatrix} x \\ y \end{bmatrix}$ is the coordinate of a point in image plane .

The process to find out R and t is called extrinsic calibration. This process has to be carried out whenever the pose of the camera or the object coordinate system is changed.

From (14), one formula which will be very useful later is derived.

$$\rightarrow \quad \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} * s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

$$\rightarrow \quad \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}^{-1} * \left(\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} * s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} - \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \right) = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad \textbf{(15)}$$

→ $$A*s-B=\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \textbf{(16)}$$

With $A=\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}^{-1} * \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$

$B=\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}^{-1} * \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$

$$\rightarrow \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} * s - \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} a_1 * s - b_1 \\ a_2 * s - b_2 \\ a_3 * s - b_3 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\rightarrow s = \frac{Z+b_3}{a_3}\,;\, X = a_1 * \left(\frac{Z+b_3}{a_3}\right) - b_1; Y = a_2 * \left(\frac{Z+b_3}{a_3}\right) - b_2 \quad \textbf{(17)}$$

A fundamental mathematical model for vision calculation is provided above. The formula represents mere theory: given a calibrated camera (the rotation matrix, intrinsic matrix and translation vector are known), a detected object (its x and y coordinates in image plane are known) and the distance Z from the camera to the object, the X and Y coordinates of the object in the object coordinate system are derived.

From this model, all necessary steps are built up

- Calibrating camera.
- Detecting object.
- Measuring height distance from camera to object.
- After finishing above steps, the application should be able to calculate X,Y coordinate of the objects using formula (17)(17) .

4.2.1 Camera calibration

In the previous part, a complete mathematical camera model is presented. It has parameters that must be dealt with in order to complete the model.
In summary, following steps need to be performed to fulfil calibration process

- Step 1: Intrinsic calibration to find intrinsic matrix and distortion coefficients. As the result, every new frame has to be undistorted before applying any algorithm later on.
- Step 2: Extrinsic calibration to find rotation matrix and translation vector in order to complete the mathematic model (14) of the camera.

4.2.1.1 Camera Calibration in Open CV

Although there are multiple ways to solve camera calibration, OpenCV uses those that require minimum calculation and work well for planar object. OpenCV used Zhang's method to calculate the focal lengths and Brown's to achieve distortion parameters [9] [12]. The process of calibration requires a set of one-to-one corresponding 3-D and 2-D points. Three-dimensional coordinates of points in object coordinate system are given in advance and corresponding two-dimensional points are detected in image. The set of points will act as input for solving (14) using Zhang and Brown's method.

A chessboard is used in OpenCV for calibration due to its easy generation and detection with known geometry.

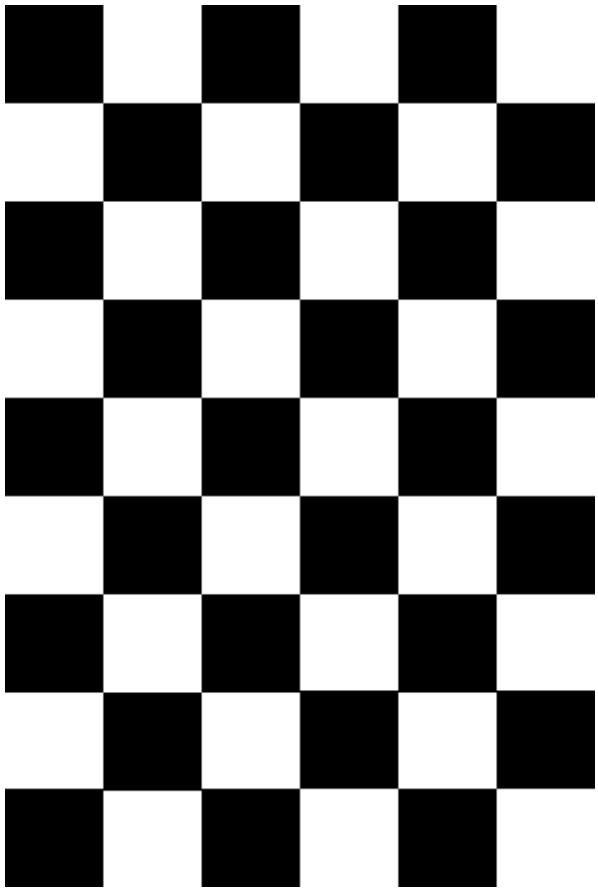


Figure 20. A simple chessboard pattern with 5x8 size.

Intrinsic Calibration

A rich set of 3-D and 2-D points is required as input for function in EmguCV [13]

```
publicstaticvoidCalibrateCamera(
        MCvPoint3D32f[][]objectPoints,
        PointF[][]imagePoints,
        SizeimageSize,
        IntrinsicCameraParametersintrinsicParam,
        CALIB_TYPE flags,
        outExtrinsicCameraParameters[]extrinsicParams
)

Parameters

objectPoints
The 3D location of the object points. The first index is the
index of image; second index is the index of the point.

imagePoints
The 2D image location of the points. The first index is the
index of the image, second index is the index of the point

imageSize
The size of the image, used only to initialize intrinsic
camera matrix

intrinsicParam
The intrisincparameters, might contains some initial values.
The values will be modified by this function.

Flags
Flags

extrinsicParams
The output array of extrinsic parameters.
```

The following method is used to obtain objectPoints and imagePoints as parameters for above function. A chessboard and a function to detect chessboard corners [13] are available

```
publicstaticboolFindChessboardCorners(
Image<Gray, byte> image,
SizepatternSize,
CALIB_CB_TYPE flags,
outPointF[] corners
)

Parameters
image
Source chessboard view

patternSize
The number of inner corners per chessboard row and column

flags
Various operation flags

corners
The corners detected
```

In order to obtain a rich set of points, the chessboard is rotated on the surface of the table to create many different views of the chessboard. Generally, more than ten poses of chessboard are needed in order to get a good result [2].

In order to get objectPoints easily with different poses, an uncomplicated object coordinate system is required. The cell dimension of the chessboard is known. The object coordinate system is defined the same as chessboard coordinate system with origin is the first corner of the chessboard, Z axis points upward, X and Y axis overlap with X and Y axis of the chessboard. Therefore, the corner 3-D coordinate can be computed with index of n

$$x[n] = \frac{n}{board_w} * cell_w \quad \textbf{(18)}$$

$$y[n] = (n\%board_w) * cell_l \quad \textbf{(19)}$$

$$z[n] = 0 \quad \textbf{(20)}$$

with

$board_w$ is width of board(in cell).

$cell_w$ is width of a cell(in millimetre).

$cell_l$ is length of a cell(in millimetre).

As the result, coordinates of every corner can be calculated easily no matter how you rotate your chessboard.

In order to get imagePoints, a new frame from camera is queried whenever the user changes chessboard pose and run the FindChessboardCorners function on the frame. The functions will return an array of found corners which are imagePoints.

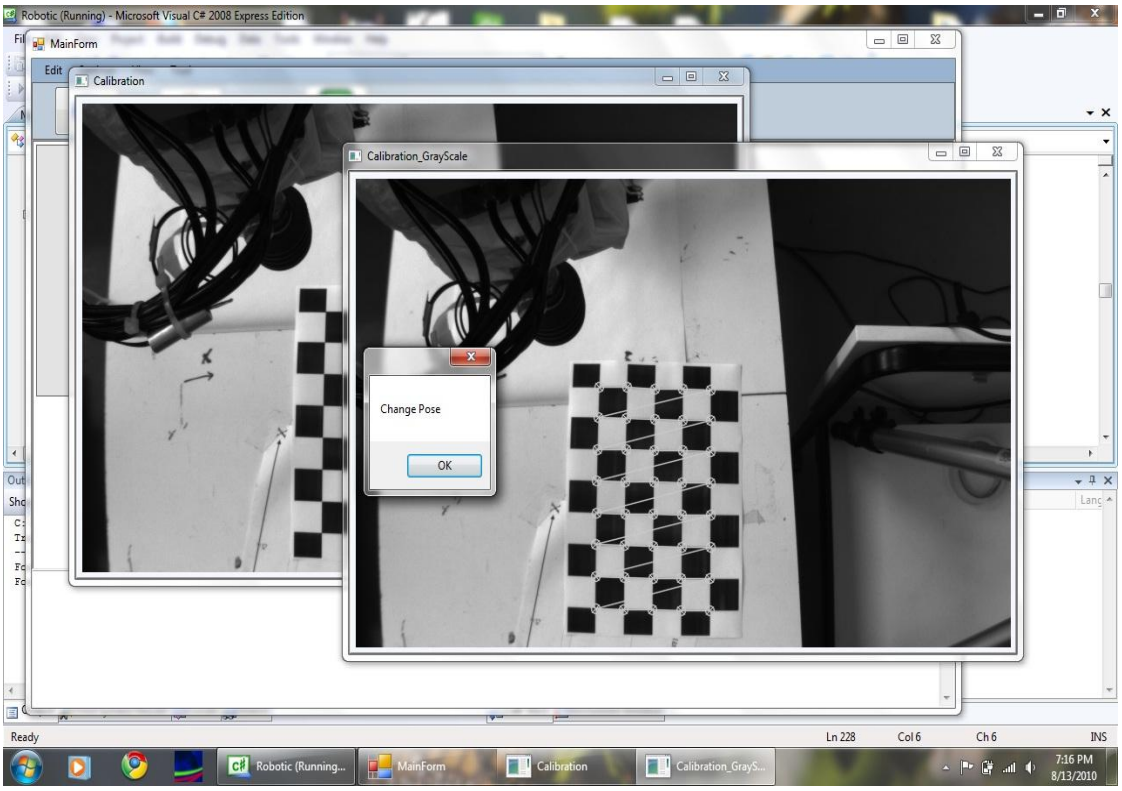


Figure 21. Intrinsic calibration. All the chessboard corners are detected and application prompt user to change chessboard pose in order to detect new set of 3D-2D corresponding points.

After having objectPoints and imagePoints, the function CalibrateCamera is used to achieve intrinsic camera parameters.

Extrinsic Camera Calibration

```
Public static ExtrinsicCameraParameters FindExtrinsicCameraParams2(
    Point3D<float>[]objectPoints,
    Point2D<float>[]imagePoints,
    IntrinsicCameraParametersintrin
)


Parameters:

objectPoints
The array of object points

imagePoints
The array of corresponding image points

intrin
The intrinsic parameters
```

FindExtrinsicCameraParams2 is used to compute extrinsic matrix using known intrinsic parameters, a set of coordinates of 3D object points and their correspondent 2D projections.

In order to use this method, the chessboard is placed on the table so that the origin of the chessboard coordinate system is overlapping with the robot coordinate system (real world coordinate system). Then, function FindChessboardCorners is used to get a set of coordinates of 3D object points and their correspondent 2D projections. The intrinsic parameters are known from previous step. As the result, all the necessary parameters are available to apply FindExtrinsicCameraParams2 to estimate extrinsic parameter and complete camera model described in formula (17).

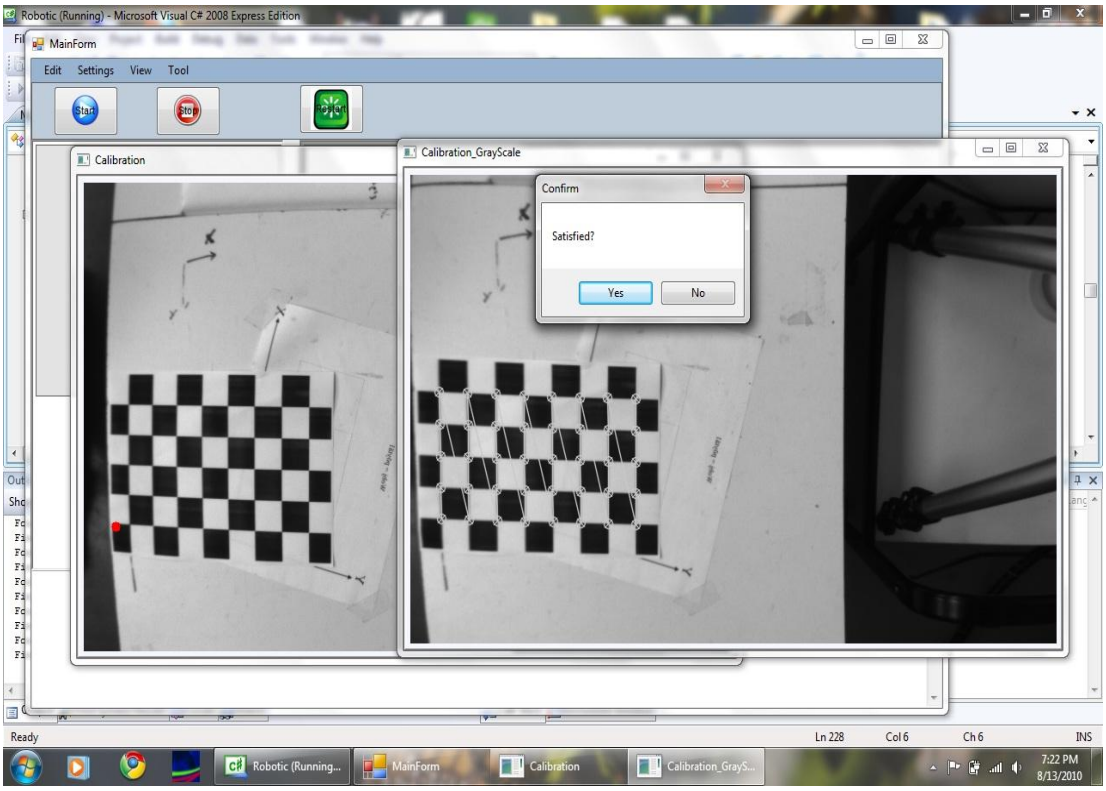


Figure 22. Extrinsic Parameter Calibration in which the chessboard is placed so that its first corner to overlap with the origin of robot's coordinate system.

The last step of the calibration process would be storing calibration parameters somewhere for later usage if the camera position or robot's coordinate system isn't changed. As C# supports serialization of objects, calibration parameters has been stored under XML (Extensible Markup Language) format. When the application starts, it will load those XML files and reconstruct C# objects without the need to do calibration again.

4.2.1.2 Calibration Result

In this application, EO-0413M 1/3" CMOS Monochrome USB Camera [25] and TAMRON CCTV Lens are used as camera. Due to the long distance and low speed of USB transmission, a special USB cable is used which can provide extra power for the camera over long distance.

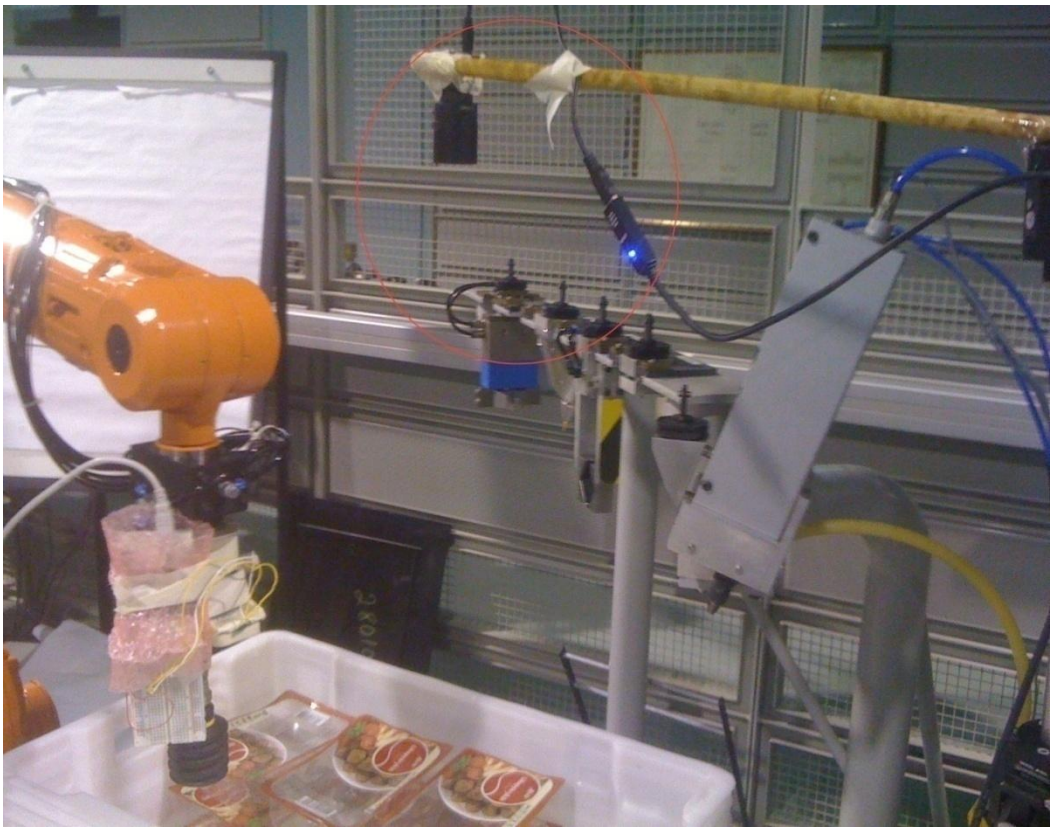


Figure 23. The camera in the working space

After calibrating, the result is as follow

Intrinsic matrix

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 939.8787 & 0 & 291.6239 \\ 0 & 949.0118 & 204.8668 \\ 0 & 0 & 1 \end{bmatrix}$$

Extrinsic matrix

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} = \begin{bmatrix} 0.0738630 & 0.966409 & -0.2461678 & -51.76666 \\ -0.99509 & 0.055149 & -0.0820756 & 186.6437 \\ -0.065742 & 0.251023 & 0.965745 & 157.20998 \end{bmatrix}$$

Distortion parameters

$$\begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} -0.453744 \\ -0.3256628 \\ -0.0071649 \\ 0.0236721 \\ 4.8843275 \end{bmatrix}$$

Apply those parameters into equations (15)

$$\begin{bmatrix} 0.0738630 & 0.966409 & -0.2461678 \\ -0.99509 & 0.055149 & -0.0820756 \\ -0.065742 & 0.251023 & 0.965745 \end{bmatrix}^{-1} *$$

$$\left(\begin{bmatrix} 939.8787 & 0 & 291.6239 \\ 0 & 949.0118 & 204.8668 \\ 0 & 0 & 1 \end{bmatrix}^{-1} * \begin{bmatrix} s*x \\ s*y \\ s \end{bmatrix} - \begin{bmatrix} -51.76666 \\ 186.6437 \\ 157.20998 \end{bmatrix} \right) = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

→

$$\begin{bmatrix} 69.41944 & 917.13448 & 219.27853 \\ -935.273 & 52.33800 & -278.97862 \\ -61.78763 & 238.22094 & 33.22007 \end{bmatrix} * \begin{bmatrix} s*x \\ s*y \\ s \end{bmatrix} - \begin{bmatrix} -199.88765 \\ -0.27157 \\ -0.27157 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \textbf{(21)}$$

This is the equation that represents the camera model at a specific pose. These testing results will be used in further analysis in other chapters.

4.2.2 Summary

In this part, the calibration process which is a primary requirement for machine vision systems is described. The process is implemented in a user-friendly and reliable method. User needs only a printed sheet of chessboard pattern and the program will do the rest of calibration, calculation and storage procedure.

4.3 Pattern Recognition

4.3.1 Introduction

This is a fast scale- and rotation-invariant interest point and descriptor. The method is built on top of the "Distinctive Image Features from Scale-Invariant Keypoints" [15] and "Speeded Up Robust Features descriptor Bay06" [20]. The aim of this method is to find the corresponding features between two images. To be more specific, this application will track in the scene for pre-defined product's existence and position. The primary requirement and challenge are speed. Speed has to be near real-time in order to compete with human speed in recognition.

4.3.2 Implementation

The aim of the approach is to extract distinctive invariant features from images that can be utilized to reliably match between different views of objects. The extracted features are invariant in terms of rotation and scale. Therefore, an object can be indentified with high robustness regardless of changes in 3D viewpoint, noise or illuminations with a rapid speed (near real time performance).

At first, a sample image of the product under .png (Portable Network Graphics) format is taken. Computer vision applications have been using this format because it supports gray scale images and does not require a patent license. Greyscale digital image has only one channel that carries the intensity information. Hence, the computer will run many times faster to process a greyscale image than a colourful image with more than one channel. Then, interest points in the image are extracted. The method to extract interest points is based on Hessian Detector to find a list of points which are invariant in terms of rotation and scale.

Figure 24. Typical greyscale png sample image.

Having a list of interest points, they will be compared against those interest points extracted from camera's frames. By comparison of interest points, it can be decided if they represent the same object points. In order to do the comparison, the interest region is divided into 4 smaller square regions. Then construct the vector for each interest region

$$v = (\sum d_x; \sum d_y; \sum |d_x|; \sum |d_y|) \qquad \textbf{(22)}$$

where

d_x is Haar wavelet transform in horizontal direction.

d_y is Haar wavelet transform in vertical direction.

At final step, this vector is normalized. The interest point comparison is now done by comparing this vector. After comparison, a list of matched features is achieved. For better accuracy, it will be voted for size and orientation to eliminate the matched features whose scale and rotation are not in harmony with the majority of them. Finally, a projection matrix (homography matrix) can be computed from those matched points to project the sample image into the frame images by using RANDSAC (RANdom SAmple Consensus).

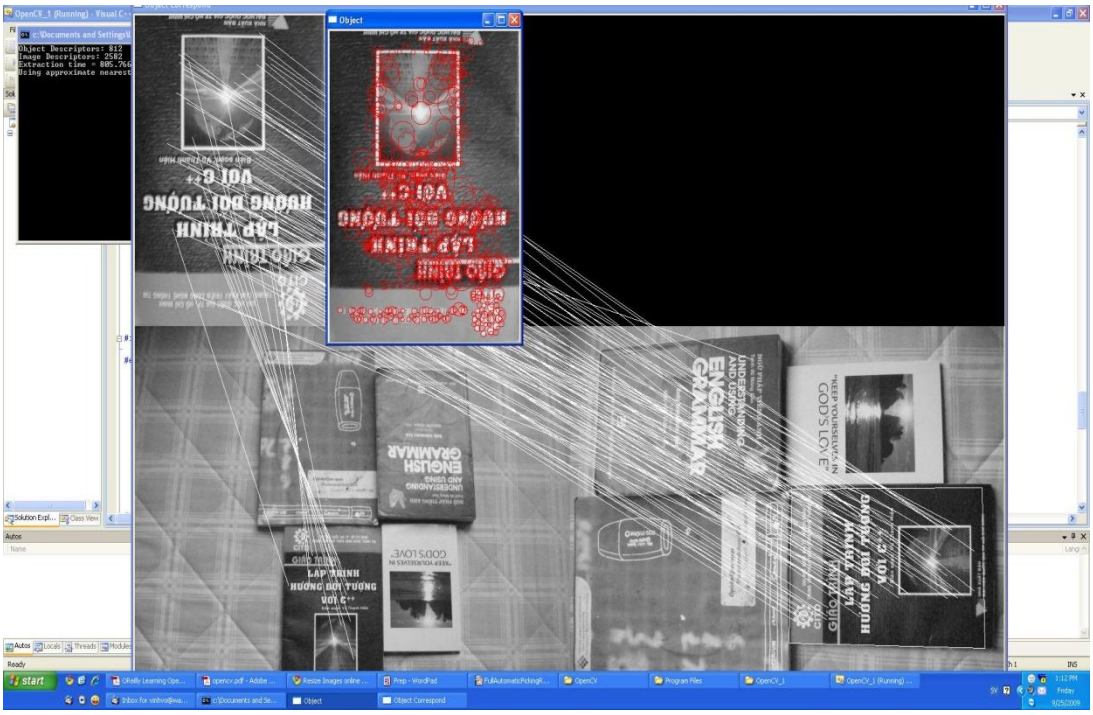


Figure 25. Interest area extraction, comparison and matching.

However, in case there are many products in the frame, the detection seems to reduce significantly the accuracy and efficiency. With one sample interest point, there are multiple matching points in the frame. RANDSAC, hence, fails to construct homography matrix. This redundancy is handled by dividing the frames into sub-regions. RANDSAC is applied for only matching points which belong to those sub-regions.

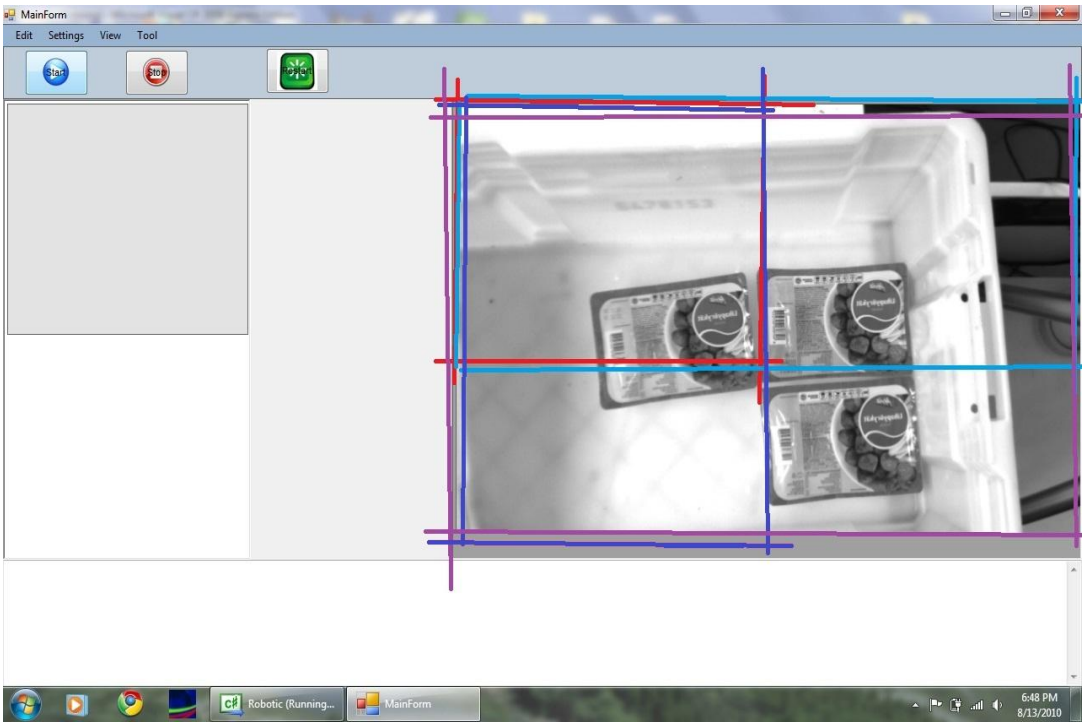


Figure 26. Sub regions in a frame. In various experiments, with the particular size of boxes, width side and length side of the frame are both divided into 2 smaller lengths. This combination gave best performance in terms of speed and accuracy.

Obviously, the application does not intend to stop after tracking one product. Recursive algorithm is brought in use to track all (or most of) products inside the box. The list of matching points obtained from comparison step is filtered out, which means matching points from found products will be eliminated. Matching points are iterated and checked if they locate inside areas of founded product. Those that locate inside areas of founded product will be eliminated. After filtering matching points, the step of dividing sub-regions is repeated and RANDSAC again. As the result, a recursive loop is executed until no more products are found.

Last, this method, however, can sometimes give bogus detections. Hence, the validation method is added to verify the existence of product. There are two criteria are used to validate the tracking result

- The projected area must not be smaller than a certain minimum value.
- The projected are must be rectangular (because all images are rectangular).

After setting parameters for above two criteria, the detection is now trustful.

4.3.3 Result of Pattern Recognition

The scene obtained from the camera as follow

Figure 27. Frame image from camera

The typical result of pattern recognition

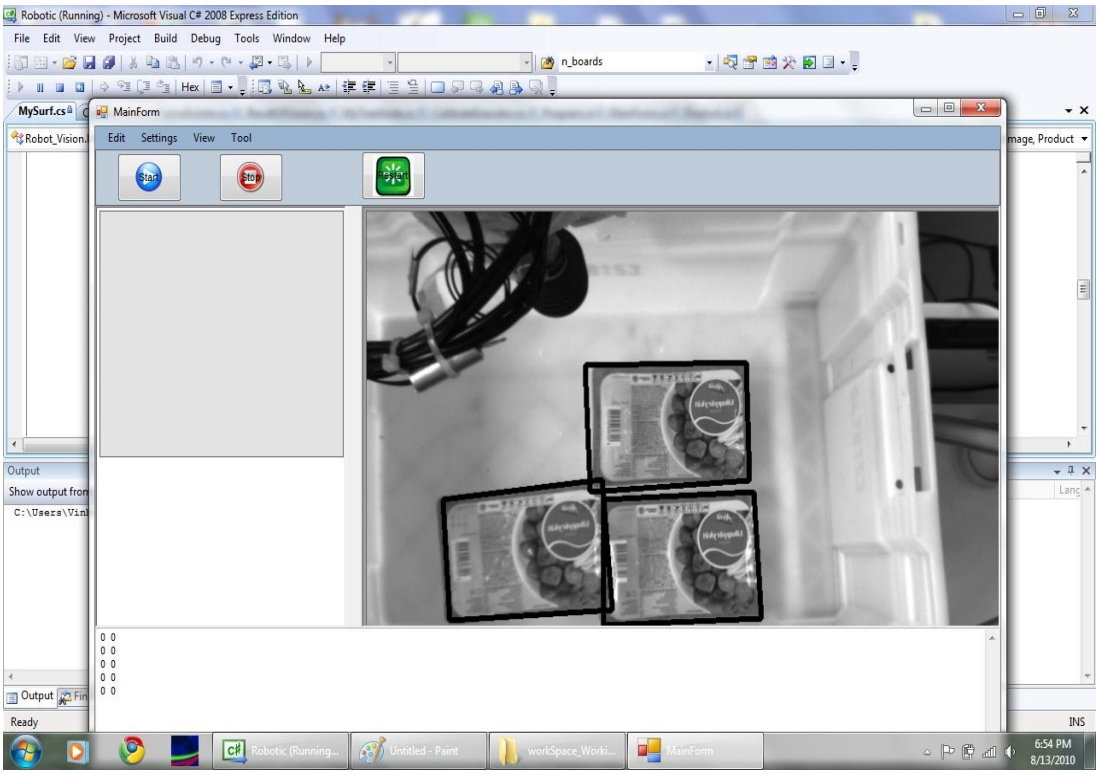


Figure 28. Apply enhanced pattern recognition.

The real-time detection is also demonstrated in test video [31] which records the screen of central computer. In the video, the application is able to track multiple products at really fast speed.

4.3.4 Summary

In this part, the method of pattern recognition has been discussed. The used method is explicitly tested with positive results. No lagging time is observed during application operation. As the result of fast tracking, robot will have constant job to perform. Usually, the vision system always has a queue of multiple products waiting robot to pick and place. Therefore, there is no wasting time in the system which means optimal speed can be achieved with faster robot movement.

4.4 Summary

In this chapter, the method of implementing the vision system is described. Mainly, it consists of two important parts which are calibration and pattern recognition. Both of described methods are implemented so that they are easy to operate and produce precise results.

5 PROXIMITY MEASUREMENT SYSTEM

5.1 Introduction

The proximity measurement system is added to complete 3D coordinate position tracking of the product. The primary requirement for the system is accuracy. Thanks to the elastic rubber part on top of the vacuum sucker, it is allowed to have 2-2.5 cm deviation. In this chapter, the method to achieve such accuracy for the system will be represented.

5.2 Implementation

5.2.1 Ultrasonic Sensor vs. Microcontroller

The proximity measurement is given by using a PING))) Ultrasonic Sensor. Parallax's PING))) ultrasonic sensor is a low-cost and rapid solution for measuring distance in various robot and security systems. Main features of this sensor

- Works by transmitting an ultrasonic pulse (above human hearing range) and output a pulse that has the time matches to the time that the ultrasonic pulse required to travel back and forth.
- Supply Voltage: 5 VDC.
- Supply Current – 30 mA typical; 35 mA max.
- Range – 2 cm to 3 m (0.8 in to 3.3 yards).
- Input Trigger – positive TTL pulse, 2 μs min, 5 μs typical.
- Echo Pulse – positive TTL pulse, 115 μs to 18.5 ms.
- Burst Indicator LED shows sensor activity.
- Delay before next measurement – 200 μs.

Figure 29. PING))) Ultrasonic Sensor [8].

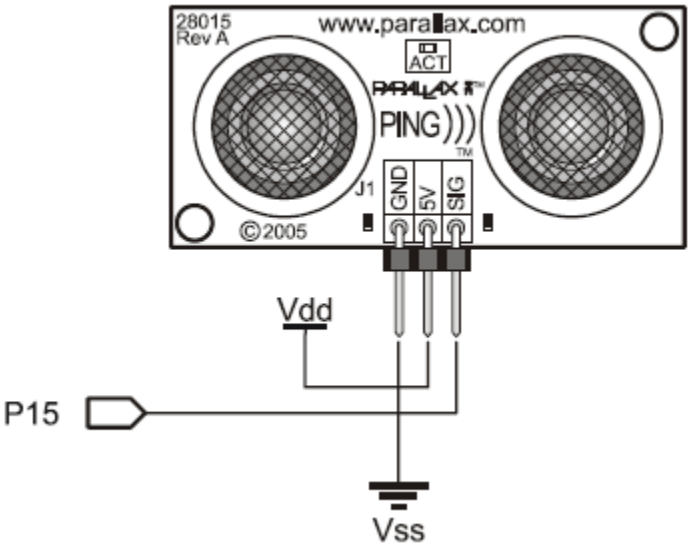


Figure 30. Quick Start Circuit for Ping))) sensor [8].

The Ping))) sensor natively can't communicate with controller computer directly. As a result, an extra microcontroller is brought to use to interface sensor with computer's serial port. The used microcontroller is Arduino which is an open-source electronics prototyping platform [5].The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) [7]. Main features of the microcontroller

Table 1. Arduino specifications

Microcontroller	ATmega168
Operating Voltage	5V (can be powered via the USB connection or with an external power supply)
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analogue Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	16 KB (ATmega168)

SRAM	1 KB (ATmega168)
EEPROM	2 bytes (ATmega168)
Clock Speed	16 MHz

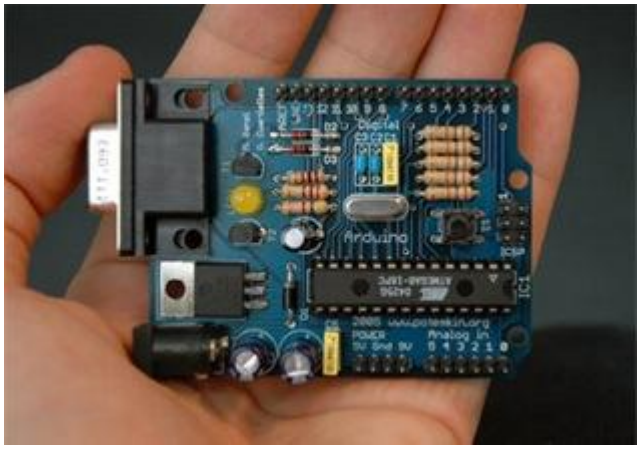


Figure 31. Arduino board [51].

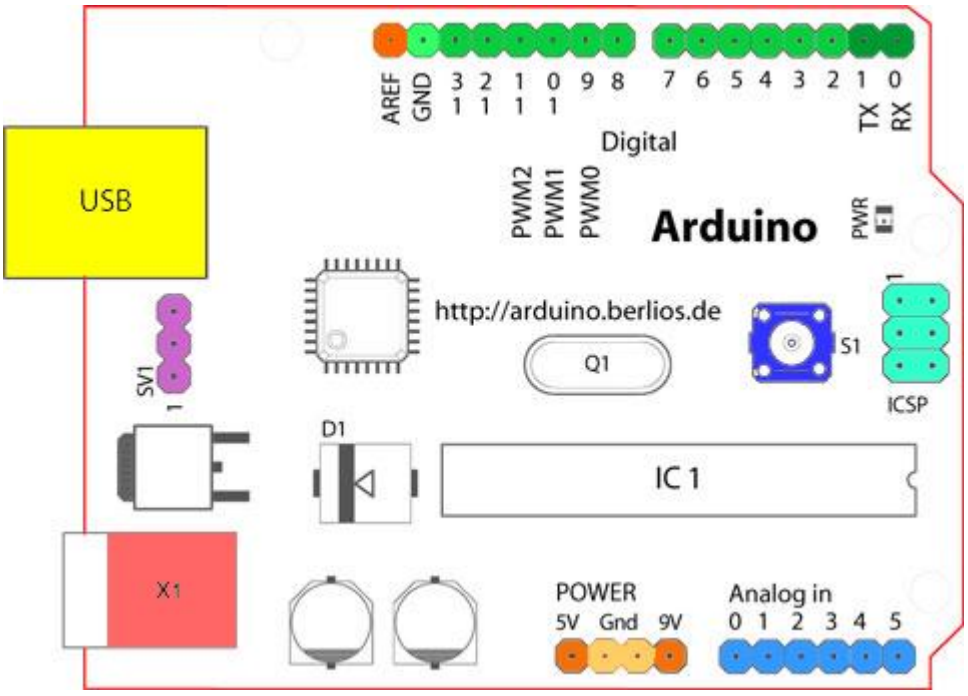


Figure 32. Outline of Arduino [5].

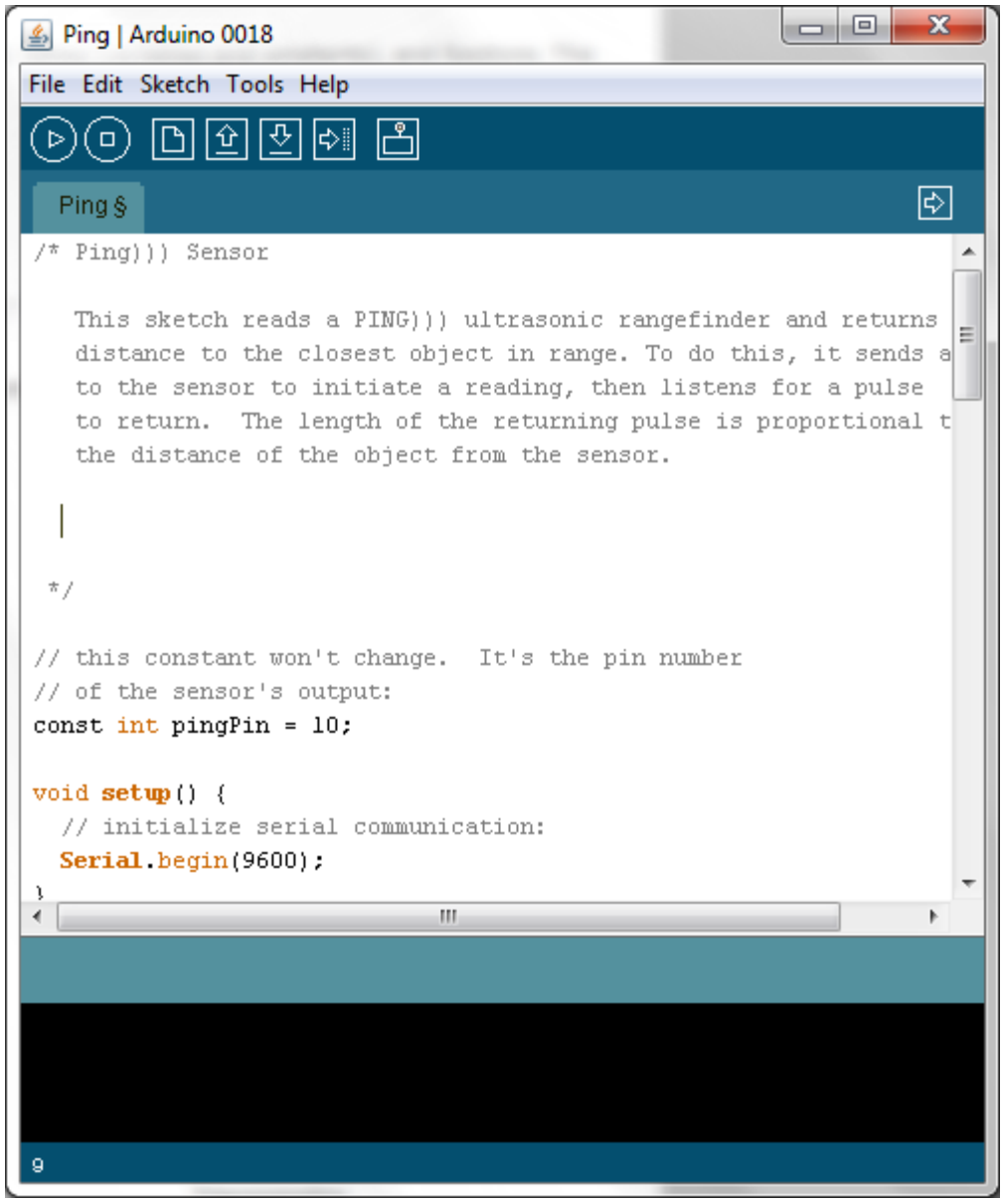


Figure 33. Arduino development environment (based on Processing) [7].

And the connection diagram is as follow

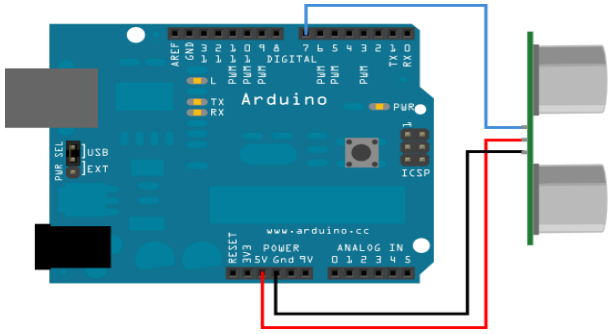


Figure 34. Arduino and Ping))) sensor connection diagram [5].

The Arduino requires a small trunk of code in order to read distance values from Ping))) and communicate with the computer.

```
//define the input signal pin in Arduino
const int pingPin = 10;

void setup() {
// initialize serial communication with bit rate of 9600
Serial.begin(9600);
}

void loop()
{
// establish variables for duration of the ping,
// and the distance result in 2 consecutive measurement in
centimeter:
long duration;
float cm,cm2;

// The PING))) is triggered by a HIGH pulse of 2 or more
microseconds.
// Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
pinMode(pingPin, OUTPUT);
digitalWrite(pingPin, LOW);
delayMicroseconds(2);
digitalWrite(pingPin, HIGH);
delayMicroseconds(5);
digitalWrite(pingPin, LOW);


 //set the pin that read the signal from Ping))) in input mode
pinMode(pingPin, INPUT);
  //read the duration of the HIGH pulse (which is the time
required by the sound signal to travel      //back and forth)
duration = pulseIn(pingPin, HIGH);
// convert the time into a distance
cm = microsecondsToCentimeters(duration);
//give microcontroller a small delay.
delay(50);

// Do again above process
pinMode(pingPin, OUTPUT);
digitalWrite(pingPin, LOW);
delayMicroseconds(2);
digitalWrite(pingPin, HIGH);
delayMicroseconds(5);
digitalWrite(pingPin, LOW);
pinMode(pingPin, INPUT);
duration = pulseIn(pingPin, HIGH);
 cm2 = microsecondsToCentimeters(duration);

// check if 2 consecutive measurement differ from each other
maximum 10%(make sure the sensor //does not give any erroneous
value ).
float percent = cm/cm2*100;
if(percent > 90 && percent < 110){
//write it to the serial channel(so that computer can read from
it)
Serial.println(cm2);
  }
delay(50);
}
```

```
floatmicrosecondsToCentimeters(long microseconds)
{
// speed of ultra sonic sound is 340 m/s or 29 microseconds per
centimeter.
// The sound signal travels back and forth, therefore we divide
the time by 2
.
return (float)microseconds / 29 / 2;
}
```

During development process, numerous experiments have been carried out to prove the accuracy and robustness of the system.

Figure 35. Testing Ping))) and Arduino.

Figure 36. Testing Ping))) and Arduino.

Output of the testing process for some typical distances

Table 2. Testing results of Ping))).

Real Distance(cm)	Measured Distance(cm)	Deviation(cm)
15	15.02	0.02
30	30.16	0.16
28	28.09	0.09
26	25.84	0.16
20	19.25	0.75
33	33.29	0.29
45	44.98	0.02
40	39.83	0.17
35	35.14	0.14
32	31.76	0.24
	Average	0.204

Sensor is mounted to the tool of robot as figure below

Figure 37. Sensor mounted to robot tool.

The challenge of using this sensor is the application has to daftly know the X, Y world coordinates of the product. The robot, hence, can move on top and start measuring the distance from its tool to surface of product. Some of proven formulas in previous chapters will be used to present the problem at this stage

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}^{-1} * \left(\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} * s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} - \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \right) = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \text{ (Formula (15))}$$

$$\rightarrow s = \frac{Z+b_3}{a_3}\,;\, X = a_1 * \left(\frac{Z+b_3}{a_3}\right) - b_1; Y = a_2 * \left(\frac{Z+b_3}{a_3}\right) - b_2$$

(Formula (17))

For used camera

$$\begin{bmatrix} 69.41944 & 917.13448 & 219.27853 \\ -935.273 & 52.33800 & -278.97862 \\ -61.78763 & 238.22094 & 33.22007 \end{bmatrix} * \begin{bmatrix} s*x \\ s*y \\ s \end{bmatrix} - \begin{bmatrix} -199.88765 \\ -0.27157 \\ -0.27157 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Those formulas, in fact, suggest "if you can track the product in image(x, y are known) and the distance Z from camera to products is known, you can solve X, Y world coordinate of products". However, Z measurement can't be performed by using this sensor system because it is required to navigate robot above product to measure Z. This problem can be solved by using "Size based range finder"

5.2.2 Size based range finder

This is the first solution to measure Z distance. The expectation is less than 1 cm deviation in measuring the distance only by using image processing. However, after various testing procedures, it is shown that it is not good enough with the deviation varies in range of [-5; +5] (cm). It finally comes in handy to provide a rough measurement of Z.

The implementation is also based on the formula (17) in chapter 4. In database, product is stored with dimension information. When one specific product is tracked, program does a loop to find the smallest difference of product size which corresponding to the right/correct distance. The pseudo code

```
realSize=real_width*real_length;
minDiff=Double.MaxValue;/*maximum value of Double.*/
realZ=0;
step=1;/*increase step of z by one*/
for(z=MIN_Z;z<MAX_Z;z+=step){/*MIN_Z and MAX_Z are constants represent
possible maximum and minimum value of Z*/
width=calX(z); /*calculate width of product based on z*/
length=calY(z); /*calculate length of product based on z*/
diff=abs(x.y-realSize)/realSize ; /*calculate the different*/
if(diff<minDiff){/*update result if the calculating size is nearer
with real product size*/
minDiff=diff;
realZ=z;
        }
    }
returnrealZ;
```

Due to the distance from the camera to the product is normally high in reference to the deviation, the accuracy will be acceptable. This distance ranges [0.5 →1] (meter) in this specific arrangement, the deviation is 5(cm) which is less than 1%. Based on formula (17), the deviation of X, Y should be very small.

After carrying out testing process, the results are as follow

Table 3. Experimental results of size based range finder.

x(mm)	y(mm)	Z(mm)	X(mm)	Y(mm)	Z(mm)	X(mm)	Y'(mm)	Deviation X(mm)	Deviation Y(mm)
25	30	775	284.91	29.0107	737	283.66	-27.365	1.2575	1.64577
68	89	775	234.00	13.0976	811	232.68	14.12	1.3217	1.0224
155	143	775	189.23	98.4423	802	191.63	100.55	2.3977	2.10766
399	258	775	87.933	361.36	740	95.04	362.57	7.1064	1.21
620	467	775	139.58	651.509	766	136.44	647.22	3.1468	4.2894
							Average	3.04602	2.055046

where

x is the X-coordinate in the image(where product is tracked).

y is the Y-coordinate in the image(where product is tracked) .

Z is the Z-distance from camera to product.

X is the X-coordinate of product in real world (achieved by using formula (17)).

Y is the Y--coordinate of product in real world (achieved by using formula (17)).

Z’ is the Z-distance from camera to product’s surface (achieved by using “size based method”).

X’ is the X-coordinate of product in real world (achieved achieved by using “size based method”).

Y’ is the Y--coordinate of product in real world (achieved by using “size based method”).

Deviation X= abs (X-X’).

Deviation Y=abs (Y-Y’).

With the average error is around [3→4] (mm), which is sufficient to make a draft calculation of X, Y and Z. Hence, the robot can navigate right above the product to take the precise distance measurement, go down and pick up the product.

5.3 Other experimental concepts

In this part, two testing concept are presented. Although these methods are unable to fulfil application’s requirements, they are worth-mentioning for their innovation and creativity.

5.3.1 IR sensor with look-up table

The architecture is to have a sensor connects to a micro-controller. This micro-controller then will connect to the computer by serial cable. The architecture remains in the final solution (the IR sensor is eventually replaced by an ultra sonic sensor). The initial sensor is a SHARP GP2D12

Figure 38. SHARP GP2D12 [20].

This IR (Infrared) distance sensor has 10 to 80 cm measuring range. Its range is very suitable for the application as it is only expected to measure within range 15 to 50 cm. Unfortunately, getting distance measurement from this sensor is not straightforward. From the specification [20], the relationship between distance and analogue output voltage is known

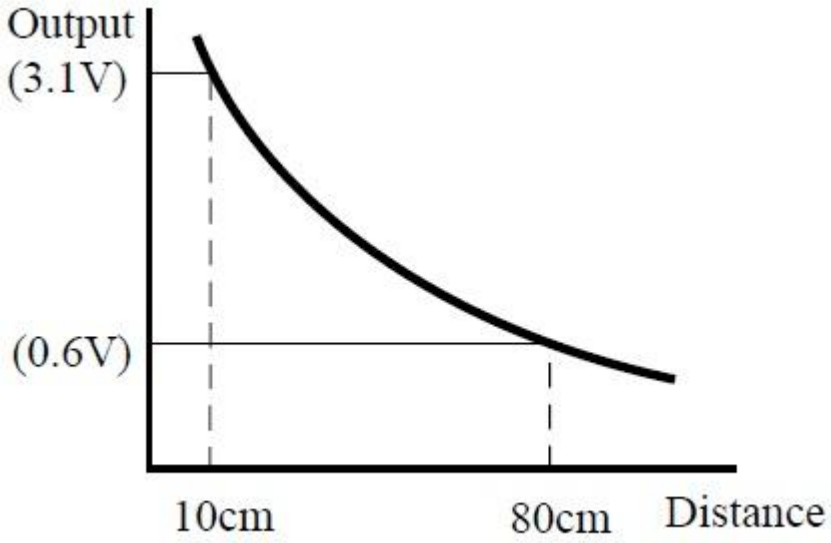


Figure 39. SHARP GP2D12 input and output relation [20].

It is clear that the mathematic relationship is not linear. As the result, given an output voltage level, there is no direct method or formula to achieve the distance. Moreover, this graph can vary from sensor to sensor. However, one good solution is to implement a look-up table. The look up table has the data as follow (for this specific sensor)

Table 4. SHARP GP2D12 experimental results.

Distance(cm)	Reading(Voltage*100)
10	223
11	214
12	205
13	197
14	189
15	184
16	178
17	173
18	168
19	164
20	161

21	155
22	153
23	149
24	147
25	144
26	142
27	140
28	138
29	136
33	132
35	128
40	124
45	122
50	113

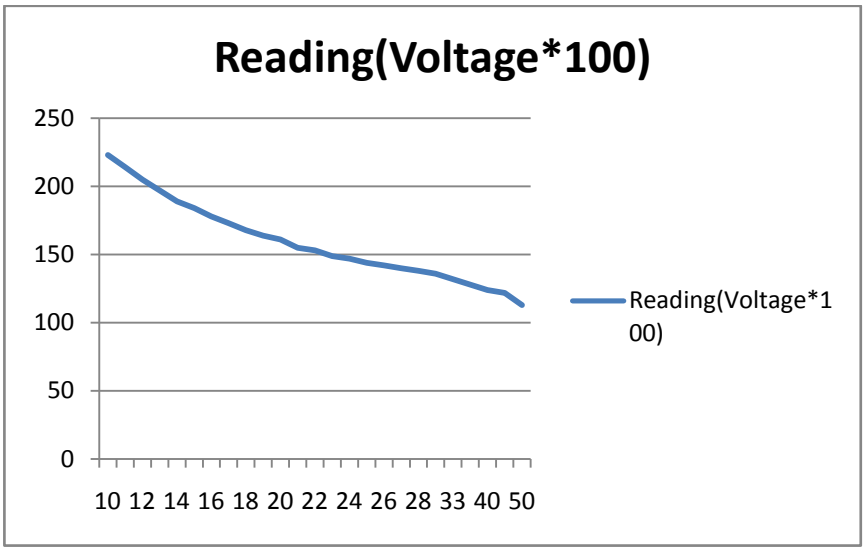


Figure 40. Input vs. Output of a specific SHARP GP2D12.

The look-up table is built base on experiment with a specific object. Each result in the look-up table is the average of three experimental results. A tape ruler is used to set a certain distance. The object then is put at measured distance from the sensor. Three results will be read from microcontroller. Finally, the average of those results is calculated and put in the look-up table. The above look-up table is achieved for a specific sensor.

When finishing building look-up table, look-up process can be performed based on basic triangle rule

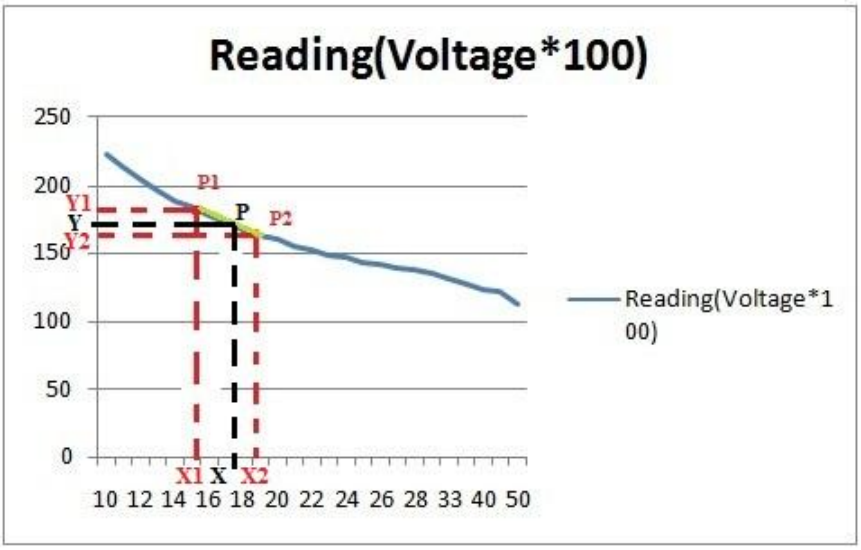


Figure 41. Look-up process.

Supposed that the reading value from the microcontroller is Y which is laid between Y_1 and Y_2. Y_1 and Y_2 correspond to pairs of distance and reading value from the look-up table (named P_1 and P_2). As in the graph above, an approximate linear method is used to get the real distance value X for the reading value Y based on simple formula by assuming P, P1 and P_2 are in the same line

$$\frac{X_1 - X_2}{Y_1 - Y_2} = \frac{Y_1 - Y}{X_1 - X}$$

$$\rightarrow X_1 - X = \frac{(Y_1 - Y_2) * (Y_1 - Y)}{X_1 - X_2}$$

$$\rightarrow \quad X = X_1 - \frac{(Y_1 - Y_2) * (Y_1 - Y)}{X_1 - X_2} \qquad \textbf{(23)}$$

As the result of this approximation, the distance value obtained from the sensor will have deviation less than 1 cm from the real values which is very satisfactory. However, when applying the method in the real environment with real product, the IR sensor seems to have disadvantages. The drawback comes from the IR itself.

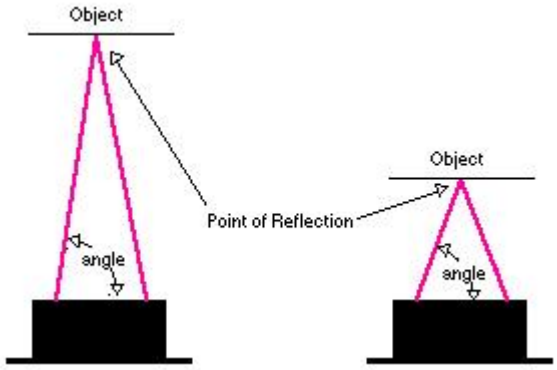


Figure 42. Different Angles with Different Distances [34].

The working principal of the IR sensor is to measure the angle of the IR light beam in order to calculate the distance. However, lights do not reflect in the same way in every surface. Therefore, the reading value will be erroneously different from surface to surface. In the application, product have plastic surface which is almost the worst case because of its highly reflective and transparent surface.

5.3.2 Laser and camera distance measurement

This is the second attempt to achieve accurate distance measurement by using a laser beam. It does sound strange, positive result is achieved with this method. With very high bright intensity, laser beam can be tracked easily by using a camera which means x, y coordinate of laser in image coordinate are known by using the formula (17) in Chapter 4.In the arrangement, X,Y of the laser generator in real-world coordinate system are fixed and known in advance. Applying those known parameters to the formula, the range (Z coordinate) from the laser generator to the obstacle can be easily achieved.

The experimental implementation is the laser generator is attached to the robot tool. Before picking products, robot has to move above the surface of the product. Obviously, the laser pointer will locate on the surface of the product. By tracking this pointer from camera, distance from the laser generator to the product can be obtained. As the result, robot will be able to navigate and pick-up the product.

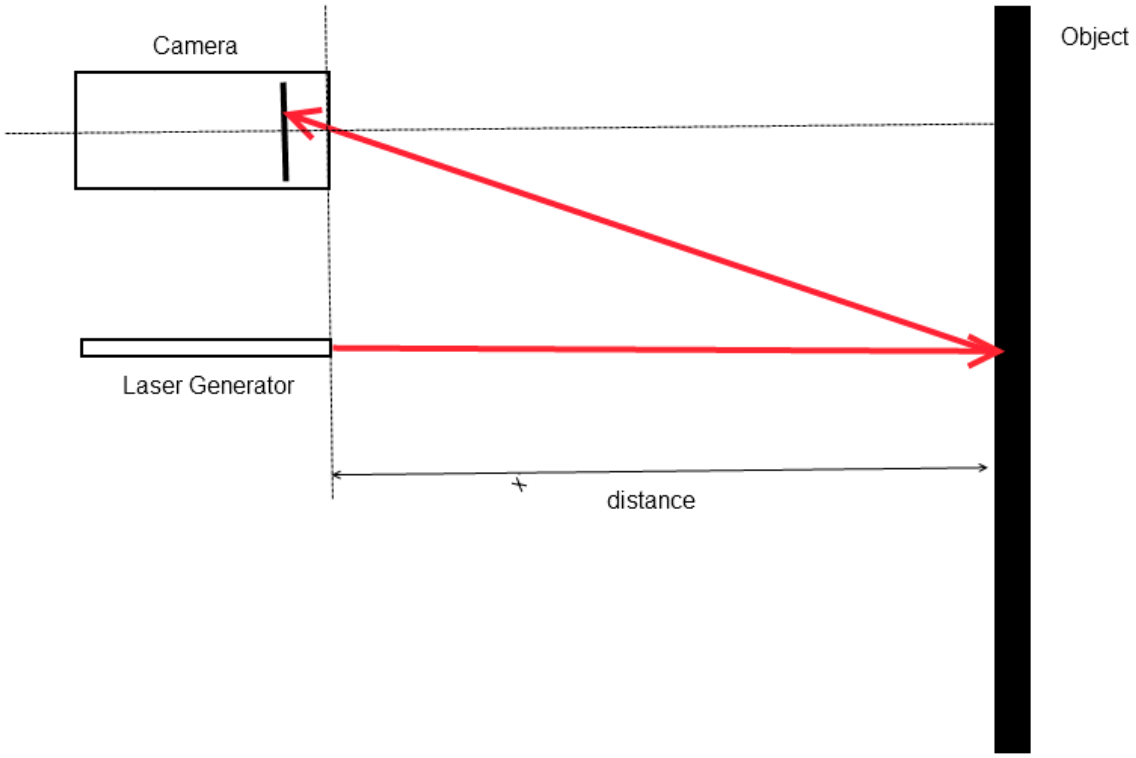


Figure 43. Laser range finder system.

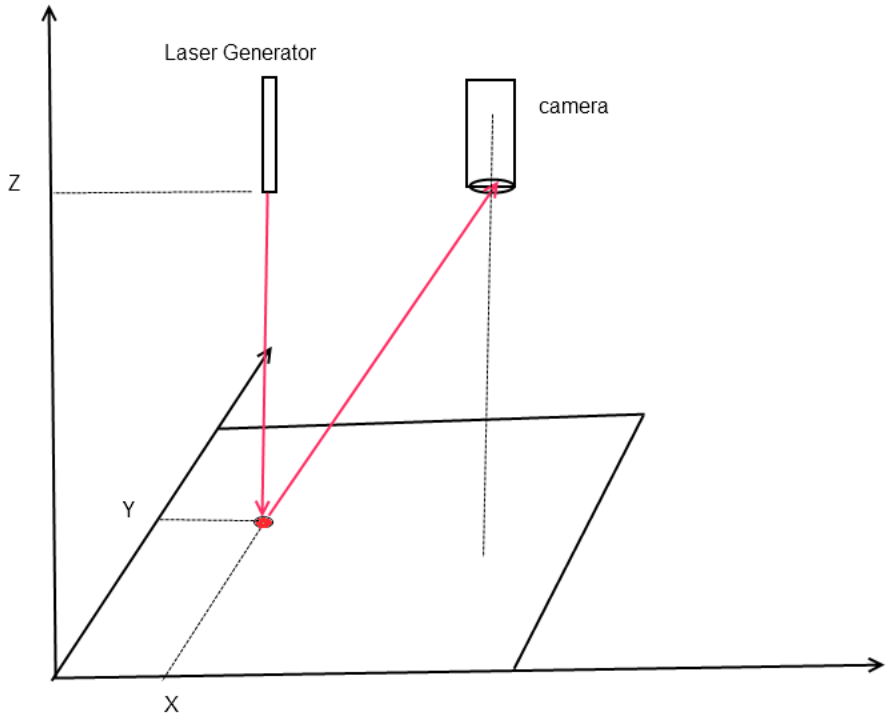


Figure 44. Laser range finder in 3-D.

This implementation will only work under certain lighting conditions in which the laser pointer will set off. A good example of optimal environment would be in a quite dark room with no direct lights or reflective surfaces. Unfortunately, it is difficult and nonstandard to build that kind of environment. Some real testes are performed in TechnoBothnia and the outcomes are not promising. Failed testes are usually caused by various environmental reasons: reflective surface of products, ability of laser beam to penetrate plastic, direct sun lights, similar colour patent on the surface of the product (redness with the help of direct sun light looks identical to laser pointer). After some seriously conducted testes, it is decided that the performance of this solution does not fulfil the requirement of industrial application.

5.4 Summary

The result achieved from experiments is very positive. Sensor's average deviation is only 0.204 cm which is well suited for this application. Especially, the results do not depend on any strict environmental condition. Thus, the combination of Ping))) and Arduino has made it possible to have a robust system to comply with high accuracy requirement.

6 BIN-PACKING SOLUTIONS

6.1 Introduction

In the application, when a product is picked from the input box, the application always has to find for it a place in the output box to drop it. Moreover, the position to drop a product always has to be optimal so that the minimal amount of container bin is used. This is a well-known mathematical problem named ”bin packing problem” [17].

The purpose of the algorithm is trying to pack objects with different dimensions into bigger fixed-size bins with minimum number of used bins. One variety of this problem is cutting material: given a big dimension material, it needs to be cut it into small pieces with highest material usable ratio. Some of real life applications are: cutting stock problem, loading truck…

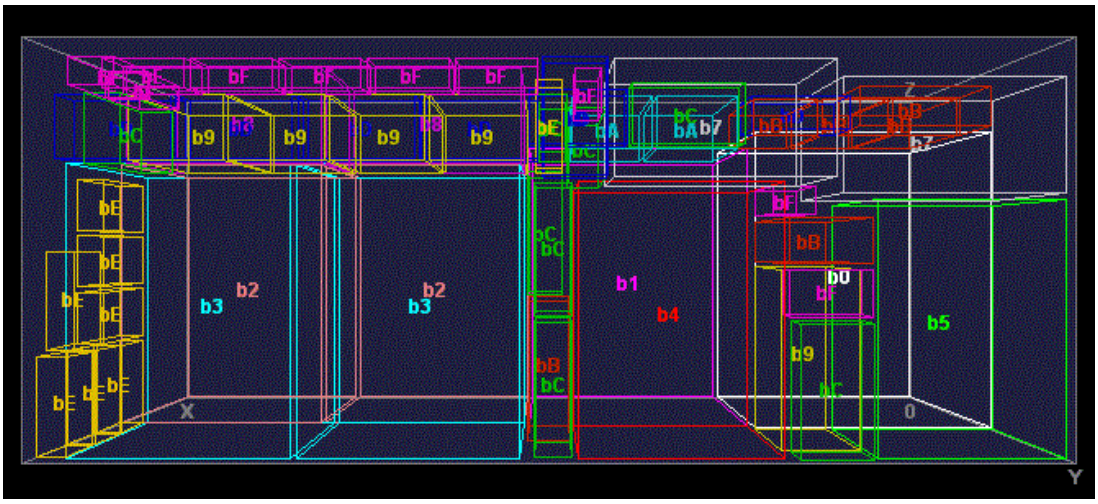


Figure 45. Bin loading problem [35].

The problem is known as "strongly NP-hard". NP-hard (non-deterministic polynomial-time hard) type belongs to the set of problem that is not sure if it can be

solved in "polynomial-time" by machine. One relevant problem has been voted as one of Millennium Prize Problems [18] is "P versus NP", better known as "polynomial-time" versus "non-deterministic polynomial-time". In which, we have to determine whether questions exist whose answer can be quickly checked, but which require an impossibly long time to solve by any direct procedure. There have been many solutions existing trying to solve the problem by using various method: heuristic, genetic, recursive...Some of the simple and famous methods are best fit decreasing and first fit decreasing. However, none of them can be claimed as perfect or best solution.

What worse is this product packaging problem even falls into the most challenging group

- It is "online": In contrast with "offline", in which, the program is given the set of objects it need to pack at the beginning and can pack them in any random order. In "online" problem, instead of having set of objects in advance, object will be given one by one. In fact, the program has the set of products it needs to pack. However, the product is not available until the vision recognizes and the robot picks it successfully. As the result, product will come in unpredictable order. Therefore, the problem turns to be “online” bin packing problem.
- The solution has to be near real-time execution which is very challenging to solve “strongly NP-hard" problems.

6.2 Implementation

The implementation is largely inspired by the method presented in “Defu Zhanga,Yan Kangb, Ansheng Denga .A new heuristic recursive algorithm for the strip rectangular packing problem”[19]. The computational results on a class of benchmark problems have shown that this algorithm not only finds shorter height than the known meta-heuristic ones, but also runs in shorter time. The average running time is $T(n)=\Theta(n3)$, which is very suitable for this application.

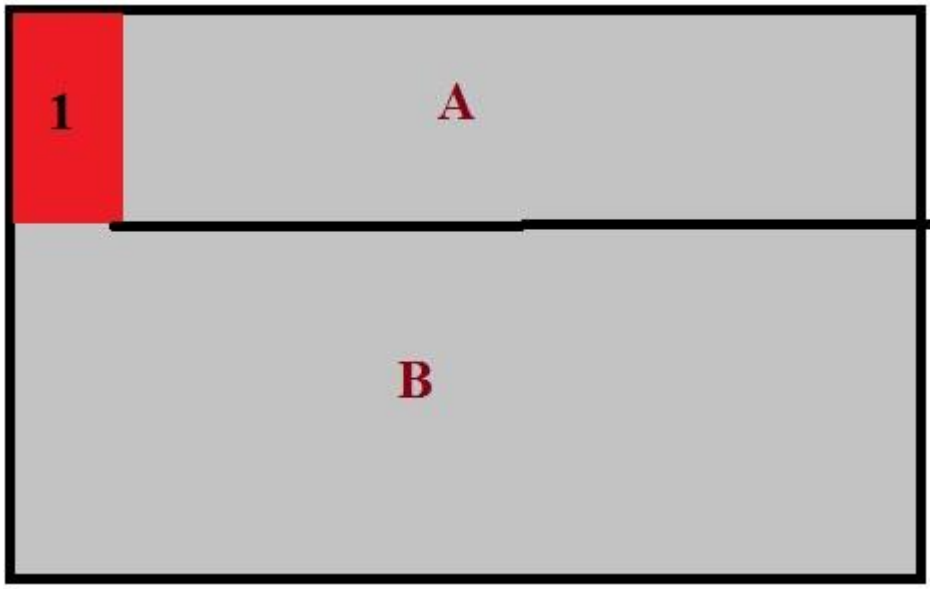
This algorithm is mainly based on heuristic strategies and a recursive structure. The theoretical method proposed in the papers is successfully implemented with some important improvements: add rotation flexibility (the robot can rotate product if it is necessary), ability to remove an added product from the structure (the picking/placing process might have problem, so the structure need to be able to cancel added product).

The recursive process will be described as follow

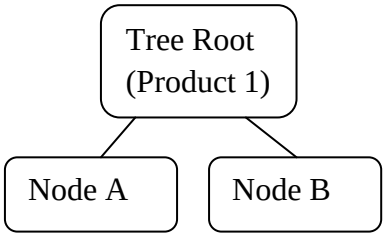
Step 1: Start with an empty box:

The tree only contains the root node.

Step 2: After inserting one product:



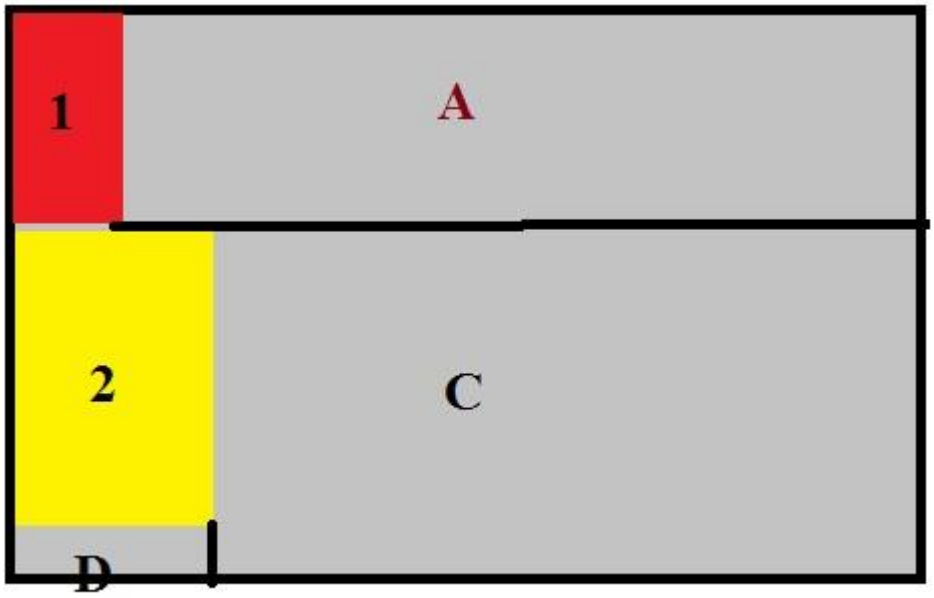
The root node of the tree now holds the first product. At the same time, the remaining free area is separated into 2 smaller areas. Therefore, the tree structure looks like this:

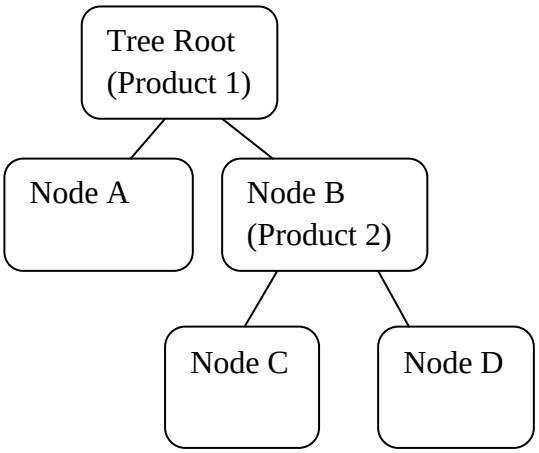


Step 3: Keep adding another product:

When inserting another product, the application first checks if the product can fit node A and if the product cannot, the application will check node B. If the product

fits into node B, it is inserted into node B and B area is divided into 2 sub-areas similar with inserting the first product.





This procedure will be repeated again and again starting from the root. If the product cannot be inserted anymore, it means that a new bin needs to be used.

Note: If neither area A nor area B can host the product, the application will try to rotate the product and insert it again. If no solution found after rotation it means that the product cannot be inserted.

There are some cases when added product is required to be removed from the tree. The application sees the product, calculates coordinates (including current product's coordinates and its coordinates in the output bin) and sends coordinates to the robot so that the robot can go, pick it up and place it to the output bin. At the same time, proximity sensor will check if the picking process has been carried out successfully. It the process fails, the bin packing application has to remove the newly added product from the tree .The newly added product is always hosted in leaf of the tree. When the product is added, it is always assigned an ascending identification number. If one inserted product needs to be removed, the application will goes recursively from the root node to find the node which hosts the product with same identification number, then set that node to null value. Because deleted node is a leaf node, it does not affect the tree structure.

6.3 Results

6.3.1 Case 1

Container bin has size of 700x300 (marked by red border)

Products: 20 products with sizes of 50x200. Occupied ratio is 95.238%.

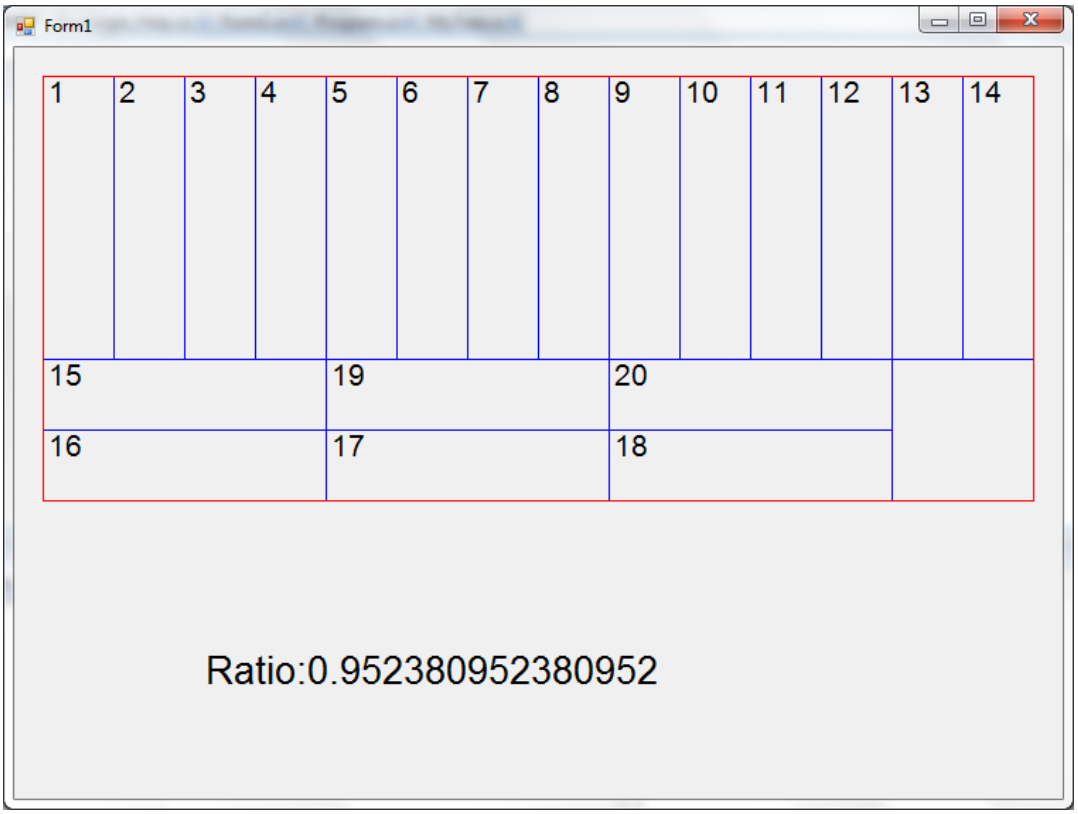


Figure 46. Bin packing experimental result.

6.3.2 Case 2

Container has the size of 700x300. Products have size of 60x100.

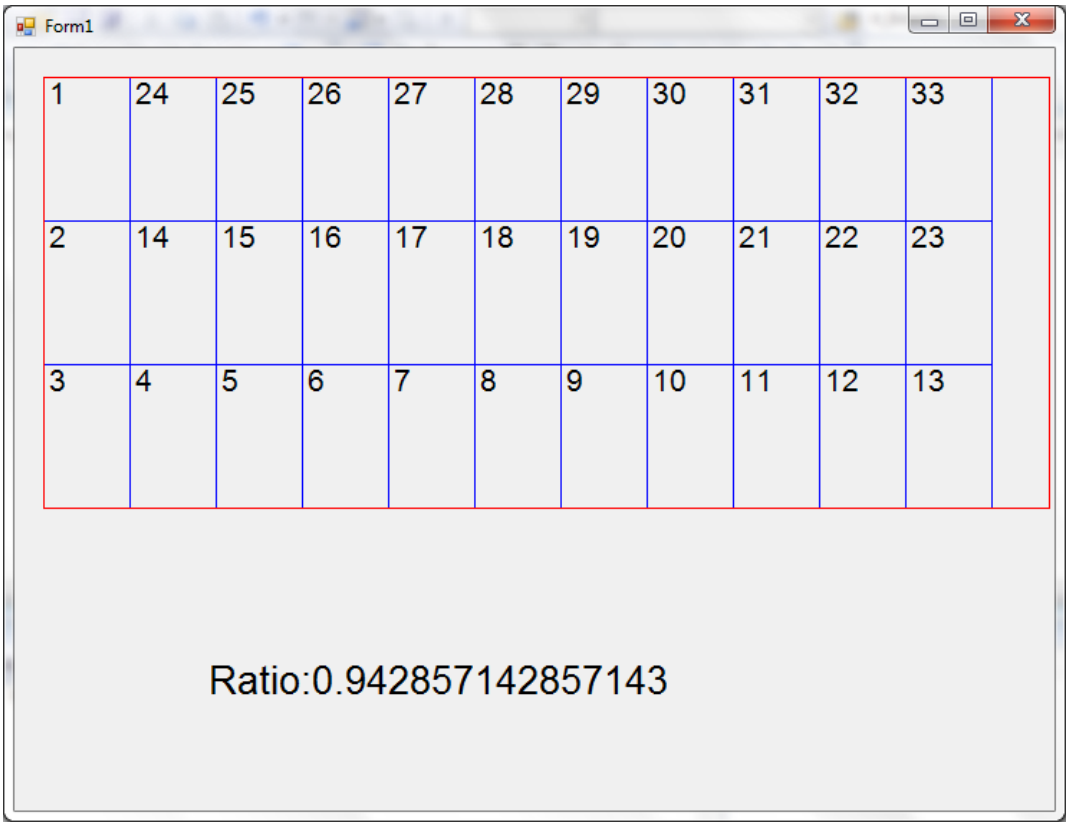


Figure 47. Bin packing experimental result.

6.3.3 Case 3

Container bin has size of 700x300. Whereas, products have random size ranging from [0 →100] x [0 →50]. The time required to execute the routine is around 4-5ms for a large set of products.

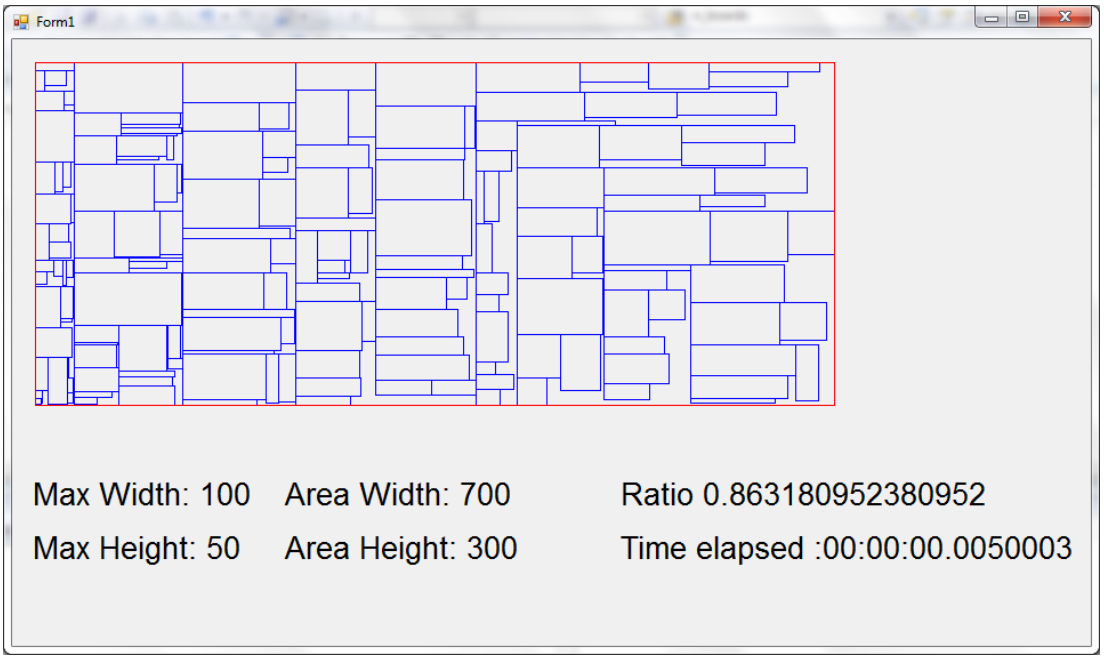


Figure 48. Bin packing experimental result.

6.3.4 More testing results

Table 5. Bin packing experimental result.

Testing result for random size boxes	
Max Width of box	200
Max Height	100
Area Width	700
Area Height	300
Number of boxes trying to insert	200

No.	Occupied Ratio
1	0.9284
2	0.93424
3	0.94944
4	0.95437
5	0.95132
6	0.94932
7	0.96235

8	0.93594
9	0.95562
10	0.93786
Average	0.945886

6.4 Summary

In this chapter, a familiar binary tree structure is implemented to solve a complex problem relatively well. Based on the test result, this solution has been proven to be good enough to solve the bin packing problem within strict timing requirement.

7 COMMUNICATION SYSTEM

7.1 Introduction

The project is the integration of many parts: software running in main computer, software running in micro-controller, software running in robot, sensor and camera. As the result, the obvious requirement is to have some system to enable the communications among separate parts. In application's design, computer plays as centre role or master, while other parts mainly communicate with computer as slaves.

7.2 Implementation

Following figures are the implementation of the system's communication

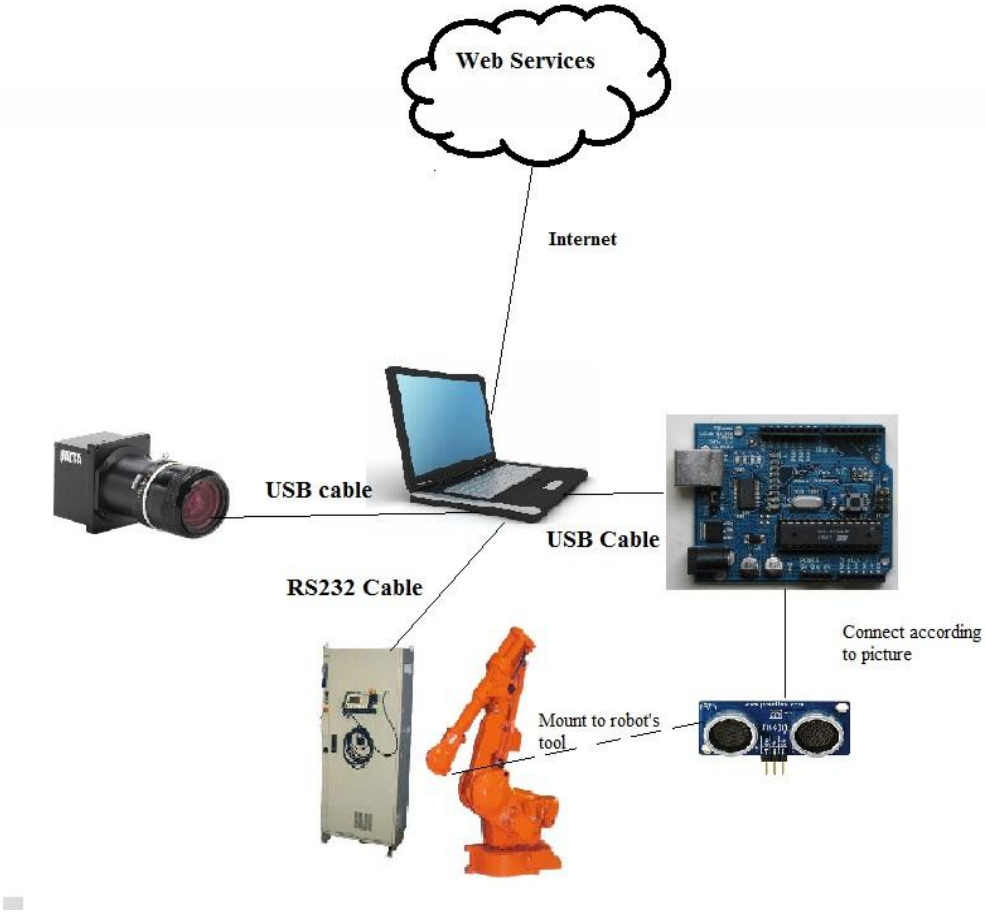


Figure 49. Connection Diagram.

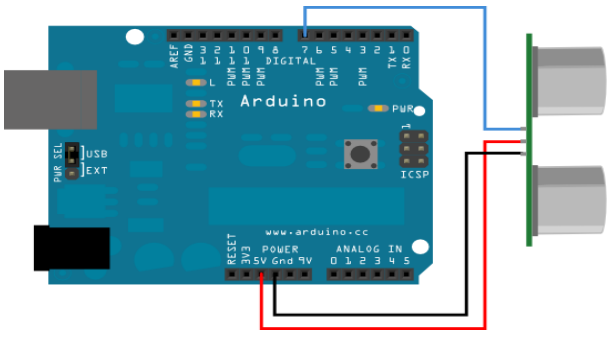


Figure 50. Connection between Arduino and Ping))) sensor [20].

The implementation consists of components as follows

- A Internet connection link to connect program to web service running at server to receive orders such as: number of product, types of product …
- An USB link to connect the camera to the computer. This connection can be replaced by IEEE 1394 interface (Firewire interface) for better transmission speed (800 Mbit/s transfer rate for IEEE 1394 versus 480 Mbit/s of USB 2.0). The camera sends all frames to the computer via this link. Camera's frames will be analyzed to track the product.
- An USB cable to connect Arduino to the computer to get data from the proximity sensor. The distance data received from Ping))) sensor will be used to navigate the robot along Z axis.
- A serial link (RS232) is used to connect the computer to the robot. This will transmit coordinate data from the computer to the robot. As the result the robot can navigate, pick and drop product.

Each communication link has to be managed within the program. The challenge is to manage many communication links, handle synchronization among sent and received data and take actions quickly when abnormal event happens. One solution has been proven efficient is described below

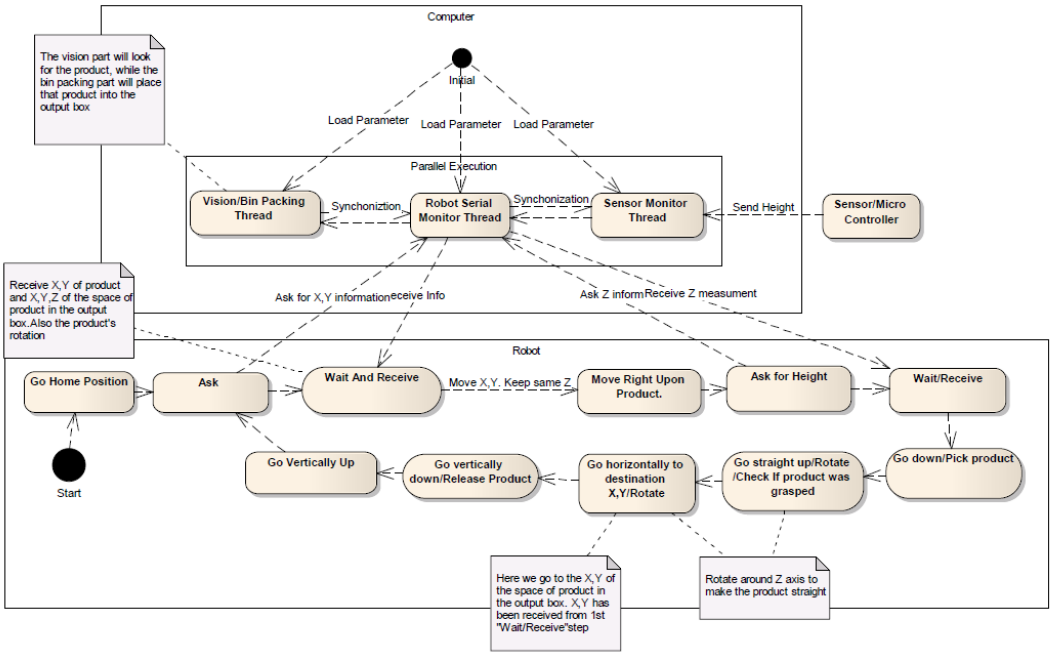


Figure 51. State Machine Diagram.

Theoretically, the host computer will be running on 3 threads

- Vision/Bin packing thread: This thread will handle the camera communication. It will detect the product using SURF and try to place the product into output box using bin packing algorithm.
- Robot serial port monitor thread: This thread will take the responsibility of communicating with the robot. It will deal with messages when robot needs more products or coordinate for navigation. Even more, it will manage all the accidents that might happen when pick and place product. Typical event is the robot picked the product and the vacuum grasper is not able to pick the product up, robot will immediately inform and demand the computer for next action.

- Sensor monitor thread: is brought in use to manage the proximity sensor data. This will keep track of the data. If data exceed a certain predefined maximum or minimum values, the program execution will be terminated. This thread is also used to check if the robot has successfully picked the product by checking to distance from the sensor to the product when the robot is going up.

7.3 Summary

The advantage of this design with several threads running in parallel is it can quickly receive and transfer data, respond quickly with events. As the result, the application achieves better performance and reliability.

8 DATABASE AND CONFIGURATION

8.1 Introduction

In this chapter, the method of using NHibernate [21] and SqlLite [22] is presented to manage configuration parameters and sample images. During the development process, an efficient solution is required to centrally manage all resources such as configuration parameters and sample images.

NHibernate is a port of Hibernate Core for Java to the .NET Framework. It is an object-relational mapping(ORM) library which provide functionality for mapping simple objects/models in object-oriented programming to database tables via XML mapping files. It is usually served as the DAO (Data Access Object) layer for applications.

SqlLite is an embedded relational database management. It has no independent process for communicating with applications. Instead, it stores entire database in a single cross-platform file. In this specific case, it is well-suited for the purpose of seeking a way to manage parameters and resources. Especially, EmguCV supports image serialization, thus, products' image set is easily managed.

8.2 Implementation

The database structure is described below.

8.2.1 CalibrationSettingConfig table

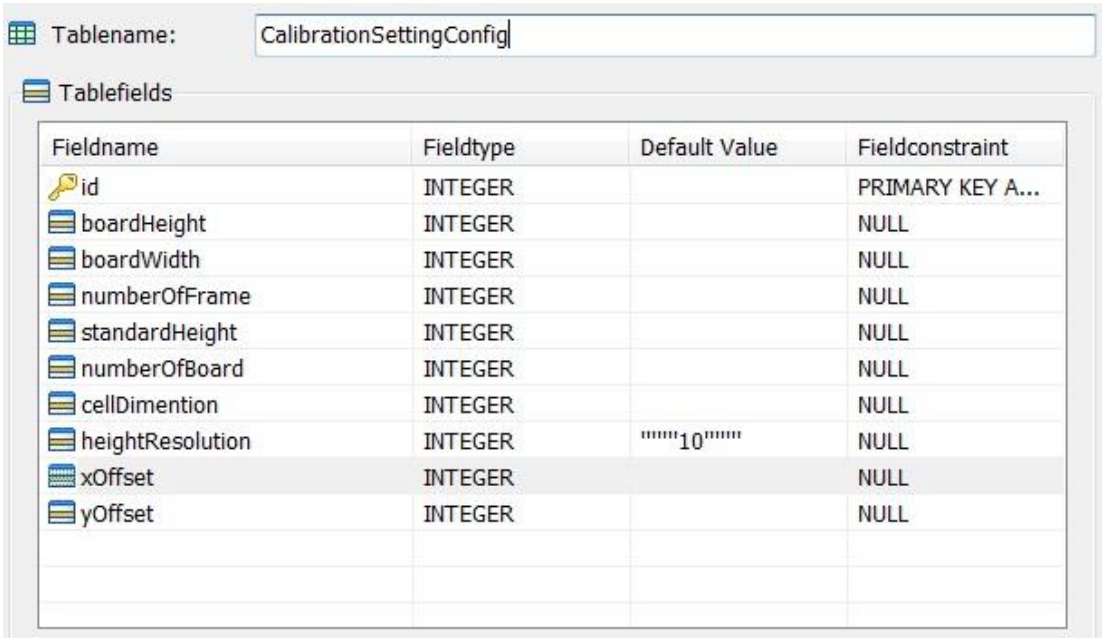


Figure 52. CalibrationSettingConfig table structure.

CalibrationSettingConfig table is used to store all parameter related to calibration process.

Table 6. CalibrationSettingConfig table structure.

Fieldname	Description
boardHeight	The chessboard length dimension in "cell" unit.
boardWidth	The chessboard width dimension in "cell" unit.
numberOfFrame	Number of delayed frames between 2

	chessboard recognition.
standardHeight	The height from camera from camera to the table where the chessboard lying.
numberOfBoard	Number of chessboard recognitions in intrinsic calibration.
cellDimention	Dimension of chessboard's cell. (The cell is in square shape).
heightResolution	This is actually the step parameter in the "sized-based" tracking method described in chapter 5.
xOffset	After doing external calibration process and observing the performance of the robot, this value can be added to or subtracted from the t_1 parameter in translation vector. Though, this parameter is not regularly put in use (default value is 0).
yOffset	After doing external calibration process and observing the performance of the robot, this value can be added to or subtract value from the t_2 parameter in translation vector.

	Though, this parameter is rarely put in use (default value is 0).

8.2.2 CameraProperties table

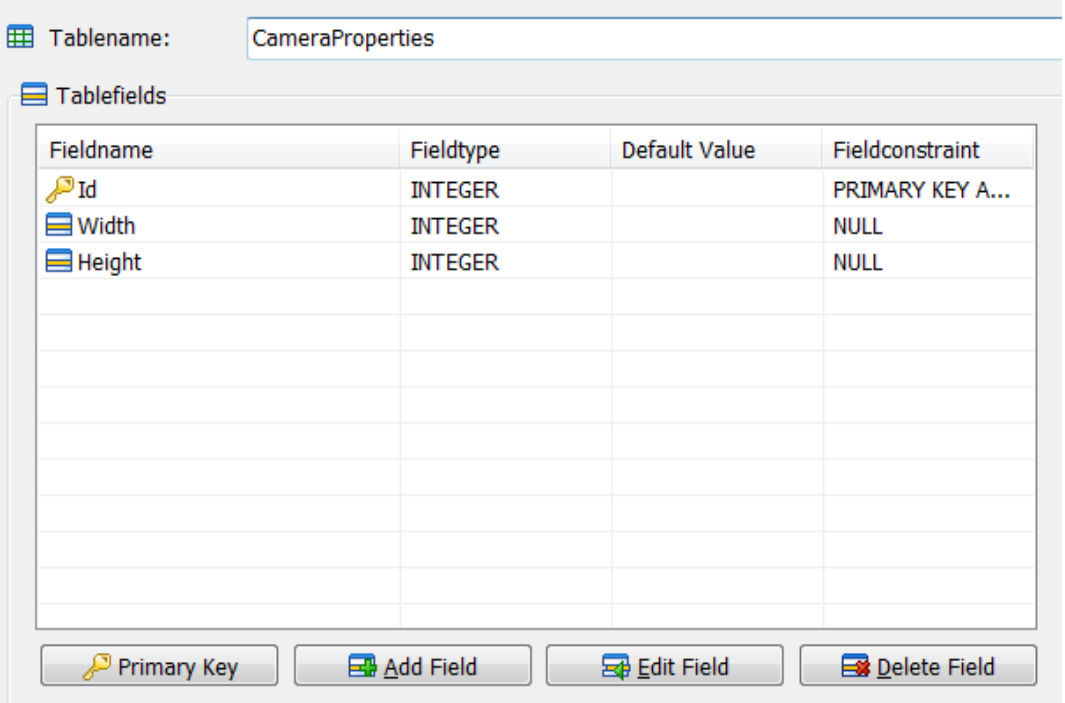


Figure 53. CameraProperties table structure.

This is used to set camera's properties. To be more specific, user will be able to set the width and the height of the camera frame (resolution). The resolution cannot be set to be greater than the maximum resolution of the camera (the camera will automatically set it to be maximum resolution). They will come in handy if you have a smaller interested area than the camera's resolution. It will reduce the pixels which the computer has to process.

8.2.3 Images table

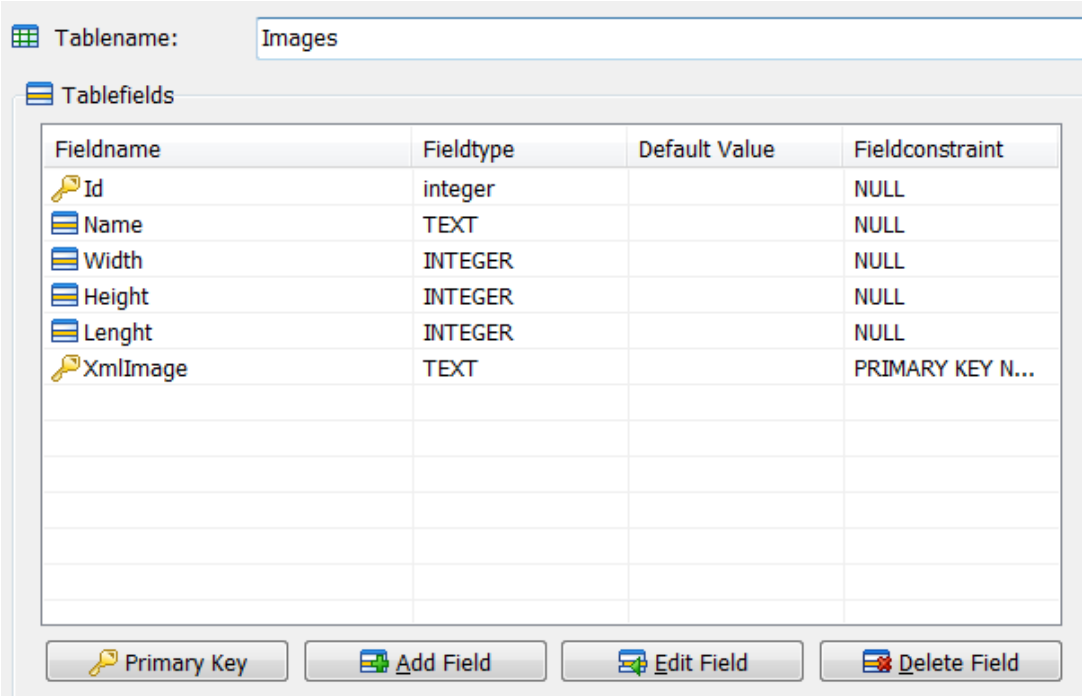


Figure 54. Images table structure.

All the products' properties will be kept in this table.

Table 7. Images table structure.

Fieldname	Description
Id	Primary key of the table.
Name	Name of the product.
Width	Width of product(in millimetre)
Height	Height of the product(in millimetre)
Length	Length of the product.

XmlImage	The product's image serialized string.

8.2.4 SURFParameterSettingConfig table

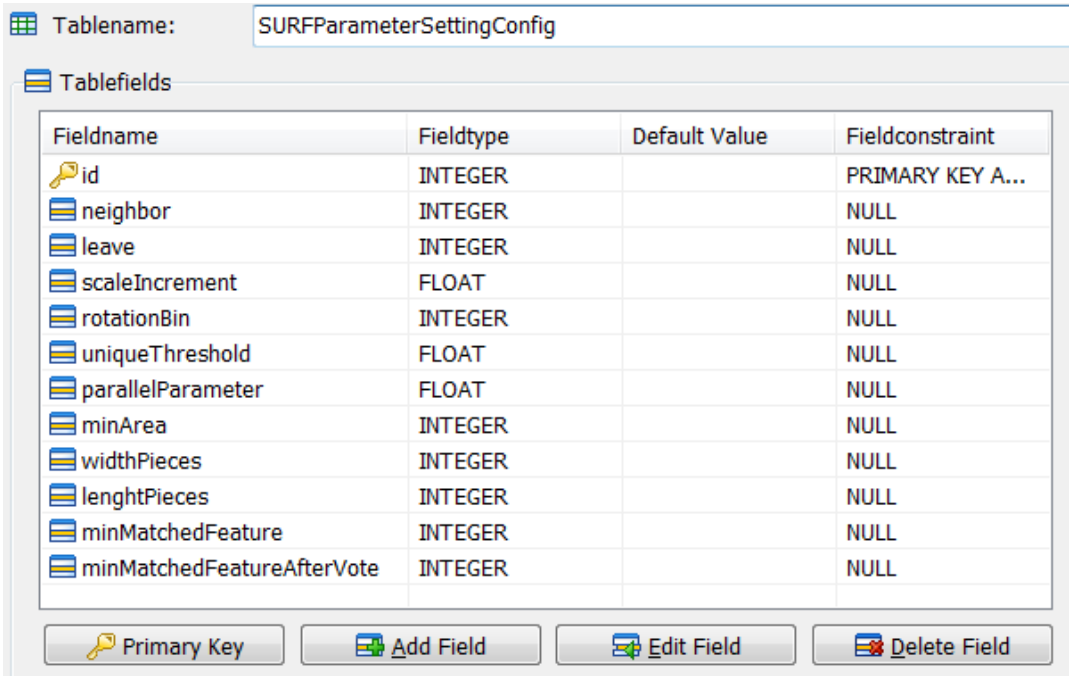


Figure 55. SURFParameterSettingConfig table structure.

The table is used to keep all the parameter for pattern tracking algorithm. All value has been set to optimal value for performance. However, in some cases, some parameters can be changed for better performance such as: uniqueThreshold, etc.

Table 8. SURFParameterSettingConfig table structure.

Fieldname	Description
neighbor	The number of neighbours to find.

leave	For k-d tree only: the maximum number of leaves to visit.
scaleIncrement	This determines the different in scale for neighbourhood bins.
uniquenessThreshold	The distance different ratio which a match is consider unique.
rotationBins	The numbers of bins for rotation.
minArea	Min area of projected area
paralellParameter	Min different between 2 length sides as well as 2 width sides to be considered rectangle shape.
widthPieces	The number of pieces that the width side of the camera's frame will be divided.
lengthPieces	The number of pieces that the length side of the camera's frame will be divided.
minMatchedFeature	The minimum number of matching features to be considered a successful tracking.

minMatchedFeatureAfterVote	The minimum number of matching features to be considered a successful tracking after vote for rotation and size.

8.2.5 WorkSpaceConfig table

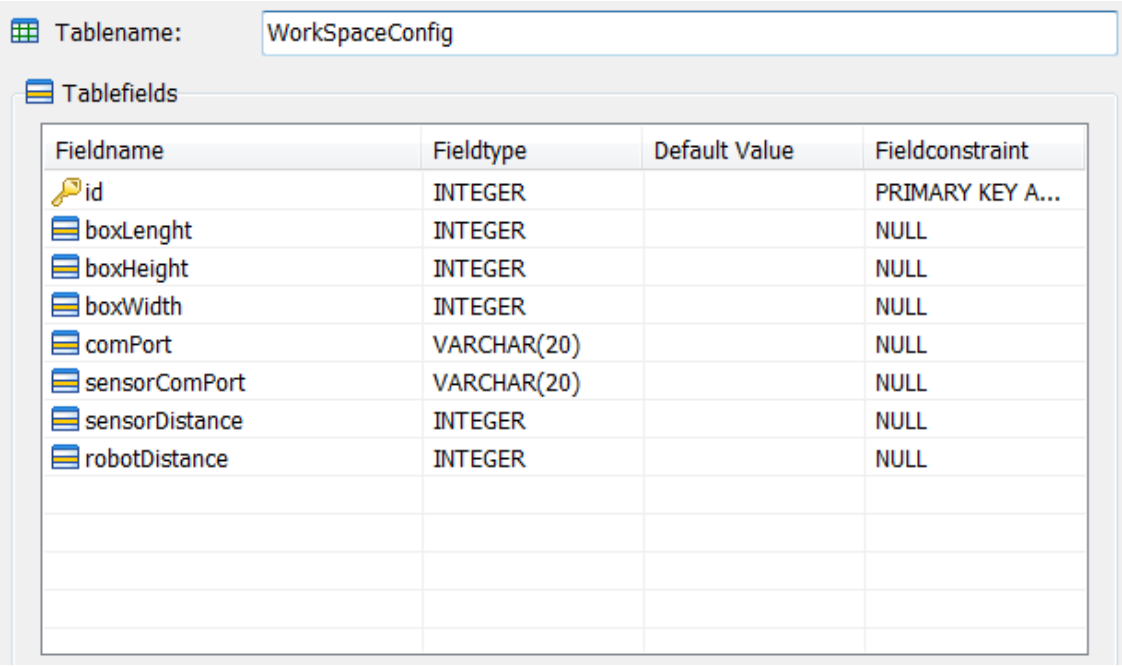


Figure 56. WorkSpaceConfig table structure.

All needed information related to the workspace is stored in this table.

Table 9. WorkSpaceConfig table structure.

Fieldname	Description

id	Primary key of the tables.
boxLenght	Container bin length.
boxHeight	Container bin height.
boxWidth	Container bin width.
comPort	Robot's communication RS232 port name.
sensorComport	Sensor's communication RS232 port name
sensorDistance	Distance from the sensor to the robot tool
robotDistance	Distance from the robot tool to the table.

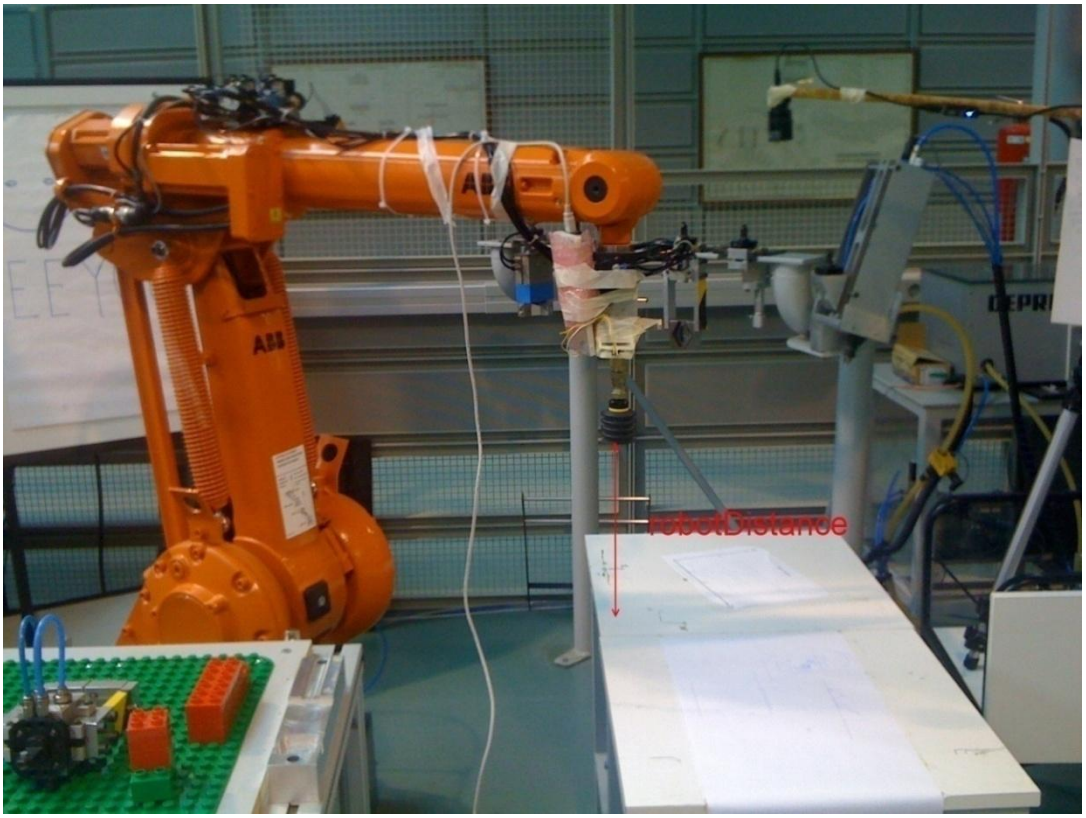


Figure 57. RobotDistance configuration illustration.

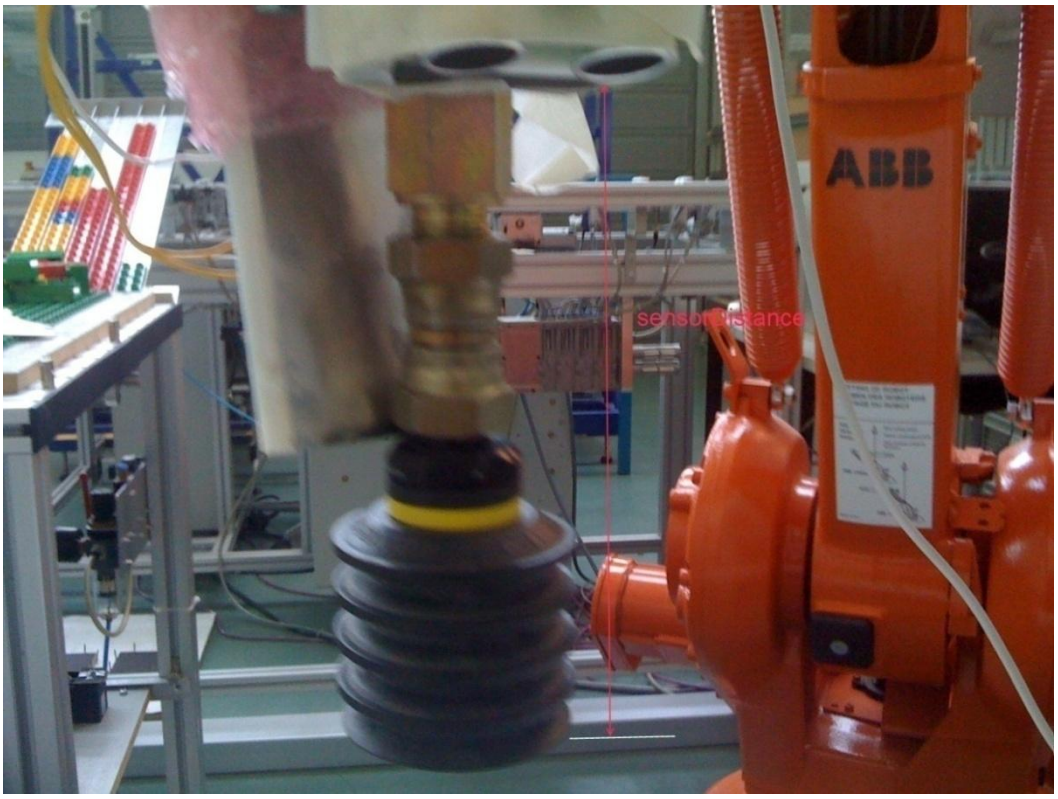


Figure 58. SensorDistance configuration illustration.

8.3 Summary

In this chapter, the database implantation and design which is also an important part of the application is described in details. The database can also be extendable easily thanks to the use of NHibernate mapping. Overall, the database side is compact, portable and extendable.

9 RESULT OF THE APPLICATION

9.1 Introduction

Multiple products and light conditions have been bought in testing process. At the end of the project, the robot is able to pick and place successfully a full box of products without the help of human with a satisfactory speed.

9.2 Result

The robot is able to navigate to complete intelligently the packaging process without human interception. This is the main criteria that the application has to fulfil [29]. Moreover, the robot can manage to handle exceptions during its work. Typical well-handled exception is failure in picking [30].

Due to the constrained speed in robot and for safety reason, the robot has never been put to operate in maximum speed. The mostly used speed which is in manual operating mode is 250 mm/s. The pace of picking and placing one product is around 8-9 seconds/product at the manual operating mode.

Moreover, some experiment to put robot to work at higher speed 600mm/s are carried out. The result is positive with 4-5 seconds/product [28]. This packing speed is really satisfactory if it can be applied it into real industrial environment.

Autonomous packaging robot vision detection testing videos are available at

http://www.youtube.com/watch?v=pzoqGo56uyg
http://www.youtube.com/watch?v=vaNgPfgL9js

Autonomous packaging robot speed testing videos are available at

http://www.youtube.com/watch?v=LShepqSAf84

http://www.youtube.com/watch?v=IsR-AHjxfzw

Autonomous packaging robot full operation testing videos are available at

http://www.youtube.com/watch?v=GtvXiN-L9L8

http://www.youtube.com/watch?v=l7IAC7yZJ_4

Autonomous packaging robot exception handler testing videos are available at

http://www.youtube.com/watch?v=3yWKlPuZsr4

9.3 Summary

The achieved speed is not as fast as human which is around 2-3 seconds per product (workers can pick and place multiple products at the same time with their both hands). Tested results have shown great potential that can be exploited from modern technology. With 4-5 seconds/product [28], if robot works constantly without "coffee" breaks, this solution definitely beats human in term of efficiency.

To conclude, the achieved result has been satisfactory and shown great potential at replacing the human worker with robotic technology.

10 CONCLUSION

This was a long-term project which required more than five months to achieve what is presented. After all, the results can be considered as a positive perspective for future development. As in chapter 9, the system is able to perform its tasks of packaging products without human interception within acceptable period of time. Therefore, the system fulfils all defined requirements at the beginning of the development process.

For continuous development, I would suggest some of possible improvements

- Building a constant lighting testing environment. The change of light will cause vision system to work inconsistently. Therefore, the time-consuming calibration process has to be done repeatedly.
- Speed up the robot to see the consequences. However, before putting robot to operate at high speed, it is compulsory to define safe regions for robot.
- Having a new tool which is thinner so that robot will not crash easily with rears of container boxes.
- There is always possibility to improve the bin-packaging solution because there is no absolute solution for this type of problem.

In brief, the system is implemented successfully. It lays the foundation for all future development and adaption. A wide range of industries is given possibility to change their way of packaging items.

APPENDIX

License of used libraries/software

Name	License Type
EmguCV	GPL or Commercial License with a small fee
.Net/C#	Freely reproduce, install and use with a Windows's license.C# can be cross-platform compiled.
SQLite	In Public Domain
Rapid	ABB robot programming language which comes with ABB robot.
NHibernate for .NET	Lesser General Public License version 2.1 (LGPL v2.1).
Arduino programming language	GPL/LGPL

REFERENCE

[1] Hynek Bakstein and Radim Halir. Camera Calibration with a Simulated Three Dimensional Calibration Object

[2] Gary Bradski and Adrian Kaehler. Learning OpenCV - O'Reilly Media.

[3] D. C. Brown. Decentering Distortion of Lenses.

[4] OpenCV Documentation. Available in www-form <URL: http://opencv.willowgarage.com/documentation/>

[5] Arduino Project. Available in www-form <URL: http://www.arduino.cc/>

[6] Wiring Project. Available in www-form <URL: http://wiring.org.co/>

[7] Processing Project. Available in www-form <URL: http://www.processing.org/>

[8] Parallax's Technology Company(Producer of Ping))) ultrasonic sensor). Available in www-form <URL:http://www.parallax.com/tabid/768/ProductID/92/Default.aspx>

[9] Z. Zhang. A flexible new technique for camera calibration.

[10] R.Y. Tsai. Metrology Using Off-the-Shelf TV Cameras and Lenses.

[11] R.Y. Tsai..An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision.

[12] Duane C. Brown. Close-range Camera Calibration.

[13] EmguCV Project Wiki. Available in www-form <URL: http://www.emgu.com/wiki/index.php/Version_History>

[14] Herbert Bay, Tinne Tuytelaars and and Luc Van Gool. SURF: Speeded Up Robust Features.

[15] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints.

[16] Microsoft .NET Framework 1.1 Redistributable EULA. Available in www-form <URL: http://msdn.microsoft.com/en-us/library/ms994344.aspx>

[17] Silvano Martello, David Pisinger, Daniele Vigo. The Three-Dimensional Bin Packing Problem.

[18] Clay Mathematic Institute. Available in www-form <URL: http://www.claymath.org/millennium/>

[19] Defu Zhanga,Yan Kangb, Ansheng Denga. A new heuristic recursive algorithm for the strip rectangular packing problem.

[20] SHARP Coop.GP2D12/GP2D15 Distance Measuring Sensors.

[21] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool .Speeded Up Robust Features descriptor.

[22] Sqlite Offical Website. Available in www-form <URL: http://www.sqlite.org/>

[23] NHibernate for .Net from JBoss Community. Available in www-form <URL: http://community.jboss.org/wiki/NHibernateforNET>

[24] Mono project. Available in www-form <URL: http://www.mono-project.com/Main_Page>

[25] Official Edmund Optics website. Available in www-form <URL: http://www.edmundoptics.com/>

[26] Technobothnia Research Centre. Available in www-form <URL: http://www.technobothnia.fi/en/>

[27] RAPID Overview. Page 295

[28] Autonomous packaging robot speed test videos. Available in www-form <URL: http://www.youtube.com/watch?v=LShepqSAf84 and http://www.youtube.com/watch?v=IsR-AHjxfzw>

[29] Autonomous packaging robot operation test videos. Available in www-form <URL: http://www.youtube.com/watch?v=GtvXiN-L9L8>

[30] Autonomous packaging robot exception handler test videos. Available in www-form <URL: http://www.youtube.com/watch?v=3yWKlPuZsr4>

[31] Autonomous packaging robot vision detection test videos. Available in www-form <URL: http://www.youtube.com/watch?v=pzoqGo56uyg>

[32] Adaptive robot programming. Available in www-form <URL: http://www.adaptiveautomation.co.uk/research.htm>

[33] ABB Arc Welding Products New Generation Positioners. Available in www-form<URL: http://www05.abb.com/global/scot/scot241.nsf/veritydisplay/d30be0b3fc0adbcdc12577890039c6b0/$File/New%20Generation%20Positioners.pdf>

[34] Sharp IR Rangers Information. Available in www-form<URL: http://www.acroname.com/robotics/info/articles/sharp/sharp.html>.

[35] Astrokettle algorithms. Available in www-form <URL: http://www.astrokettle.com/>